FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

XR interoperable immersive storytelling authoring tool using virtual choreographies

Tiago André Monteiro Ribeiro

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Fernando Cassola

Second Supervisor: Prof. Maria Van Zeller

July 17, 2025

XR interoperable immersive storytelling authoring tool using virtual choreographies

Tiago André Monteiro Ribeiro

Mestrado em Engenharia Informática e Computação

July 17, 2025

Abstract

The rapid development of Extended Reality (XR) technologies has opened new pathways for immersive storytelling, particularly in educational and cultural heritage contexts. However, existing tools for creating such experiences often require technical expertise and are limited to specific platforms. This thesis addresses these challenges by presenting a no-code authoring tool that leverages virtual choreographies to enable non-technical users to create and deploy immersive narratives across both Augmented Reality (AR) and Virtual Reality (VR) environments. Rooted in the Design Science Research Methodology (DSRM), the development process involved iterative refinement, incorporating a node-based narrative model and a choreography-driven abstraction layer to ensure XR interoperability. The proposed tool allows users to define high-level interactions and story goals independently of platform-specific constraints, significantly enhancing accessibility and reusability. The developed prototype revealed virtual choreographies' potential to democratize immersive content creation, empowering educators, curators, and cultural stakeholders to design meaningful experiences without programming knowledge. This work contributes to the fields of immersive storytelling and cultural heritage education by bridging technical gaps through an interoperable and semantically-driven authoring framework.

Keywords: Immersive Storytelling, No-Code Authoring Tool, Virtual Choreographies, XR Interoperability, Augmented Reality, Virtual Reality, Cultural Heritage Education

ACM Computing Classification System:

- Human-centered computing → Human computer interaction (HCI) → Interaction paradigms → Mixed / augmented reality
- Human-centered computing → Human computer interaction (HCI) → Interaction paradigms → Virtual reality
- Human-centered computing → Human computer interaction (HCI) → User interface design
- Human-centered computing → User centered design
- Human-centered computing → Interaction design → Systems and tools for interaction design

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) define a global agenda for achieving a more inclusive, equitable, and sustainable future. Although primarily technological in nature, this thesis contributes to multiple SDGs by promoting access to immersive educational tools, reducing barriers to cultural storytelling, and fostering digital inclusion through no-code interfaces.

The most relevant SDGs addressed in this work include:

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

SDG 11 Make cities and human settlements inclusive, safe, resilient, and sustainable.

SDG	**Target**	**Contribution**	**Performance Indicators and Metrics**
4	4.7	Supports accessible, experiential learning on cultural heritage topics through immersive storytelling tools.	Number of educational narratives created and deployed using the tool in informal or formal contexts.
	4.a	Provides educators with low-barrier XR authoring tools, reducing the need for technical expertise, upgrading education facilties.	Ratio of non-technical users successfully authoring and deploying experiences.
9	9.c	Enhances access to information and communication technologies by enabling narrative creation without programming.	Usage rates by non-developers and deployment on diverse XR platforms.
11	11.4	Strengthens efforts to protect and promote cultural heritage by making immersive storytelling tools available to museums, educators, and curators.	Case studies or partnerships with cultural institutions using the tool to share site-based narratives.

Through its emphasis on accessibility and support for cultural education, the tool developed in this thesis demonstrates how digital innovation can align with the broader mission of the United Nations Sustainable Development Goals.

Acknowledgements

First of all, I would like to thank my parents for their unconditional support throughout my life. Their encouragement and belief in me have always been my foundation, and without them, none of this would have been possible.

I am especially grateful to my grandparents, the ones who have been by my side throughout the last five years. Their presence, care, and wisdom have helped me stay grounded, even during the most demanding times of this journey.

To my aunt Maria João, thank you for always being available to help whenever I needed it, especially during the time I spent away from home. Your support made the distance much easier to bear.

I would also like to express my deep gratitude to my supervisors, Professor Maria Van Zeller and Professor Fernando Cassola, for their guidance, availability, and insight throughout this dissertation. Your mentorship played a crucial role in shaping this work.

Finally, to all the colleagues who were part of this journey, thank you for every project and every exam. Facing those challenges together is what gave this path its meaning.

Tiago Ribeiro

"Tell me and I forget, teach me and I may remember, involve me and I learn."

Xunzi

Contents

List of Figures

List of Tables

List of Acronyms

XR	Extended Reality
VR	Virtual Reality
AR	Augmented Reality
DSRM	Design Science Research Methodology
UI	User Interface
UX	User Experience
API	Application Programming Interface
JSON	JavaScript Object Notation
3D	Three-Dimensional
LMS	Learning Management System
SCORM	Shareable Content Object Reference Model
GLB	Binary GL Transmission Format (used for 3D models)
GPS	Global Positioning System

Chapter 1

Introduction

1.1 Context

The rapid development of XR technologies (VR, AR, MR) has transformed the way immersive environments are used for education and training, a trend that began with NASA's virtual simulations in the 1990s. Modern platforms like Oculus, HoloLens, and Apple Vision Pro have enabled new forms of interaction with digital content. (Cassola and Sousa [13])

At INESC TEC, the concept of Virtual Choreographies was introduced, offering a higher-level semantic approach to defining user actions, enhancing the flexibility and richness of immersive experiences (Lacet, Penicheiro, and Morgado [29]). Recent projects, such as the VESTAS wind turbine training system (Cassola et al. [14]), have demonstrated the potential of these technologies in training and educational settings. This research aims to develop an authoring tool that integrates Virtual Choreographies and XR technologies, enabling the creation of personalized, immersive storytelling experiences set within the historical framework of a medieval castle.

1.2 Motivation and Research Objectives

The use of virtual and augmented reality for education purposes is a revolutionary new step in the area, launching the public into a whole new dimension of immersion, but too often current tools for creating these immersive experiences are technically demanding and inaccessible to non-technical users such as educators, curators, or historians (Quah and Ng [40]). While no-code authoring tools have emerged in recent years, findings from the literature and market review conducted in this work suggest that these tools are typically confined to single platforms and lack the semantic abstraction required to enable true cross-platform storytelling.

This project addresses that gap by introducing a novel approach based on virtual choreographies, a high-level semantic framework that enables users to design immersive stories in terms of abstract interactions and goals, rather than platform-specific commands or device-dependent inputs.

Choreographies depict a set of acts, including behaviors, interactions, and occurrences, that are performed by actors, with well-defined goals and boundaries (Silva, Silva, and Morgado [48]). This means that rather than focusing on low-level actions or technical inputs, virtual choreographies describe these actions in terms of their higher purpose or impact. For example, in an immersive storytelling context, a virtual choreography could describe an interaction like "exploring a castle" or "inspecting a medieval artifact" as a whole action, rather than breaking it down into specific movements or gestures from the user.

According to Cassola et al. [15], virtual choreographies must also allow for multi-user operations and be independent of the platform, actor, and scenario. This independence makes them highly flexible and adaptable, ensuring that the same choreography can be used across different virtual or augmented reality platforms, regardless of the device or user actions. In other words, a virtual choreography designed for one system can seamlessly be implemented in another, allowing for a broad range of interactions and collaborative experiences in varied immersive environments.

With this tool, the contribution to the conservation and education of cultural heritage will be accessible to a wider audience.

An exploration of both the literature and the current market landscape revealed that, although AR and VR authoring tools do exist, many require specialized programming knowledge. This presents a significant barrier for non-technical users, such as educators and cultural heritage professionals who wish to create immersive content without technical expertise. Furthermore, existing no-code tools, while promising, tend to be platform-specific and lack the semantic abstraction needed to support truly interoperable experiences.

To the best of my knowledge, there is currently no widely available authoring tool that enables non-technical users to design immersive stories that function seamlessly across both AR and VR environment. This research seeks to address that gap by developing an interoperable, no-code authoring tool that leverages virtual choreographies to allow users to design and share immersive, personalized stories. By separating narrative intent from platform-specific implementation, this approach aims to make immersive storytelling both more accessible and more adaptable across diverse devices and contexts.

This research will focus on answering the following question:

> ***How can an interoperable, no-code authoring tool be designed to enable non-technical users to create immersive storytelling experiences for both AR and VR platforms?***

In order to answer it, clear objectives for the tool were put forward:

- Interoperability: Ensuring that the tool can represent virtual environments using virtual choreographies across AR and VR platforms under the same data source
- Making the experience creation no-code: to make the tool accessible for educators, it should require little or no technical knowledge to be able to create virtual choreographies

- Enabling the visualization of immersive storytelling across both augmented reality and virtual reality platforms in real-time.

1.3 Methodological Framework

To address the research question and achieve the objectives defined above, this project adopts the Design Science Research Methodology (DSRM), as described by Peffers et al. [36]. This methodology provides a structured framework for tackling real-world problems by designing and iteratively refining functional artifacts - in this case, a no-code, interoperable XR authoring tool.

As illustrated in Figure 1.1, the DSRM framework is supported by three interrelated cycles, as outlined by Hevner [24]:

- **Relevance Cycle:** Connects the research to real-world needs by identifying practical problems and validating solutions in authentic use cases.
- **Rigor Cycle:** Grounds the work in theoretical knowledge, building upon and contributing to the academic foundations of immersive storytelling and interaction design.
- **Design Cycle:** Involves the iterative creation, testing, and refinement of the artifact to ensure it meets both functional and usability requirements.

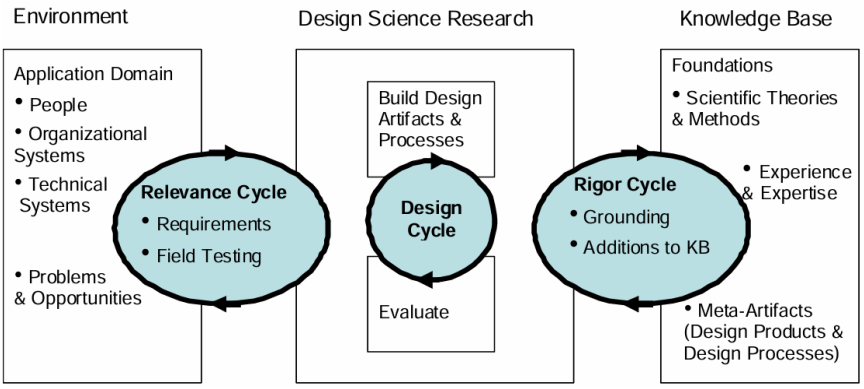

Figure 1. Design Science Research Cycles

Figure 1.1: Design Science Research Cycles by Hevner [24]

The DSRM process unfolds in a series of well-defined phases (Figure 1.2), beginning with the identification of the problem and a review of existing solutions. These insights guide the design and implementation of successive prototypes, each evaluated against the project's core objectives. While full user evaluation was not feasible within the project timeline, the methodology remains central in structuring and justifying design decisions throughout the development.

This DSR-based approach enables a balance between scientific rigor and practical utility, and it structures the remainder of the dissertation, especially the design and development chapter, where the methodology is applied through iterative refinement of the tool.

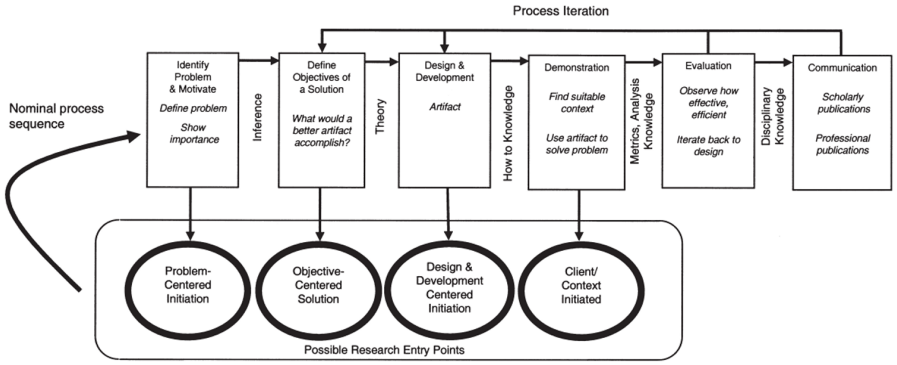

Figure 1.2: DSRM Process Model by Peffers et al. [36]

Each stage of the methodology (problem identification, knowledge grounding, artifact construction, and iterative refinement) was carried out in alignment with the goals of creating an accessible, interoperable XR authoring tool.

- **Problem Identification and Exploration (Relevance Cycle):** The research process began with a wide exploratory review across multiple areas—immersive storytelling, no-code design tools, and XR interoperability. Relevant literature was retrieved using academic databases, followed by a snowballing strategy where key references within those works were also reviewed. This phase informed a deeper understanding of the practical and theoretical gaps in the field.

- **Knowledge Foundation (Rigor Cycle):** The insights obtained from the exploratory review and literature analysis were used to build a structured state of the art. This helped define core requirements for the system, grounded in both user needs and best practices from the field. Previous work by Freitas [20] also provided a concrete foundation and served as a reference point for comparison and enhancement.

- **Artifact Construction and Evaluation (Design Cycle):** Based on the findings, an initial prototype was developed to expand on the existing editor and incorporate new capabilities. Feedback from supervisors and early evaluations led to a refined architecture that better addressed platform independence and authoring usability. A second prototype, based on a manifest-driven three-file architecture, was then incrementally built to reflect these changes. The process followed DSR's iterative build-and-refine model.

The detailed implementation of these development iterations and design decisions is described in chapter 4. The DSR methodology provided the structure necessary to align practical tool development with research objectives, ensuring that each refinement cycle addressed both functional requirements and theoretical contributions.

1.4 Document Structure

This dissertation is organized into six main chapters, each aligned with the Design Science Research Methodology (DSRM) adopted in this work.

- **Chapter 1 – Introduction:** Presents the context, motivation, and problem that led to this research. It introduces the concept of virtual choreographies and defines the research question. The methodology used to guide the development process is also described here.
- **Chapter 2 – Literature Review:** Describes the systematic review process used to explore prior work related to immersive storytelling, no-code authoring tools, interoperability, and virtual choreographies. It also details the search strategies, selected works, and key findings that form the theoretical basis for the project.
- **Chapter 3 – State of the Art:** Analyzes trends, gaps, and design choices in current academic research and commercial tools. The review is organized by thematic axes and includes a market overview to contextualize the proposed tool.
- **Chapter 4 – Development:** Follows the iterative process defined by DSRM, documenting three main design cycles. Each iteration presents challenges, proposed solutions, and architectural refinements, culminating in a no-code authoring tool that supports cross-platform immersive storytelling via virtual choreographies.
- **Chapter 5 – Discussion and Evaluation:** Reflects on the work done, outlines requirements that were not achieved, and discusses future evaluation plans. It also presents scenarios where user testing could provide valuable insight into the tool's usability, effectiveness, and flexibility.
- **Chapter 6 – Conclusions:** Summarizes the main contributions of the research, acknowledges its limitations, and suggests directions for future work, including improvements to the interface, expanded functionality, and platform abstraction strategies.

Chapter 2

Literature Review

2.1 Exploratory Review

The exploratory review aimed to provide a foundational understanding of key concepts and terminology relevant to this research. By reviewing my predecessor's work, namely, "Authoring tool for multiplatform interactive digital storytelling" by Cardoso [12] and "Enabling Co-Creation for Augmented Reality: A User-Friendly Narrative Editor for Cultural Heritage Experiences" by Freitas [20], but also reviewing work on virtual choreographies, namely, "Representation of virtual choreographies in learning dashboards of interoperable LMS analytics" by Costa [17], insights were gained into various aspects of immersive storytelling, no-code authoring tools, and virtual choreographies. These documents offered valuable perspectives on the challenges of designing tools for non-technical users, the potential of choreographies to simplify complex interactions, and the limitations of existing platforms in achieving cross-platform interoperability. This review also helped establish effective search terms and refine the inclusion criteria for the subsequent literature review, ensuring a more focused and comprehensive exploration of relevant studies.

2.2 Structure

To structure this research, relevant "axes" were defined, each representing a key dimension of the study. An axis consists of one or more terms that express the same idea, frequently encountered in academic literature.

Keywords were selected to encompass the core concepts, including:

- Virtual Choreographies
- Immersive storytelling
- Interoperability
- No-code authoring tools

These keywords and their synonyms were used to form a comprehensive query structure. Tools like Scopus and Zotero facilitated document retrieval and organization.

2.3 Search Strategy

Initially, a single query combining all axes yielded no results due to excessive specificity. The search strategy was adapted into separate queries focused on each dimension, targeting studies published post-2020 and written in English for consistency.

2.3.1 Immersive Storytelling and No-Code Authoring Tools

```
TITLE-ABS-KEY("immersive storytelling" OR "AR" OR "VR" OR "Digital Storytelling"
AND TITLE-ABS-KEY("no-code authoring tool" OR "GUI-based tool" OR "Visual editor
AND TITLE-ABS-KEY("survey" OR "review") AND (PUBYEAR > 2020)
```

This yielded 1 result.

2.3.2 Interoperability in Immersive Storytelling

```
TITLE-ABS-KEY("AR" OR "VR" OR "XR")
AND TITLE-ABS-KEY("interoperable" OR "Cross-platform" OR "Modular")
AND TITLE-ABS-KEY("survey" OR "review") AND (PUBYEAR > 2020)
```

This query yielded 43 results.

2.3.3 Virtual Choreographies

```
TITLE-ABS-KEY("Virtual Choreographies" OR "Virtual choreography")
AND (PUBYEAR > 2020)
```

This yielded 6 results.

2.4 Selected Works

After removing duplicates, the corpus consisted of 49 documents. Titles and abstracts were screened, and 3 papers were deemed most relevant:

- Cassola et al. (2022): "Design and Evaluation of a Choreography-Based Virtual Reality Authoring Tool for Experiential Learning in Industrial Training."
- Marques et al. (2024): "Data Representation with No-Code Augmented Reality Authoring Tools."
- Masneri et al. (2023): "cleAR: An Interoperable Architecture for Multi-User AR-Based School Curricula."

Relevant dissertations were also included:

- Costa (2023): "Representation of Virtual Choreographies in Learning Dashboards of Interoperable LMS Analytics."
- Freitas (2024): "Enabling Co-Creation for Augmented Reality: A User-Friendly Narrative Editor for Cultural Heritage Experiences."

2.4.1 Findings and Implications

The selected works collectively highlight current research efforts in simplifying immersive content creation and enabling platform-agnostic interaction design. Three central insights emerged:

- The potential of virtual choreographies to structure interactions and narrative logic at a semantic level.
- The rise of no-code tools as a bridge between technical systems and creative users, particularly in educational and cultural domains.
- The challenge of achieving interoperability across XR platforms, with limited support for cross-device content representation in existing tools.

These themes validate the relevance of this dissertation's proposed solution and underscore the lack of integrated approaches that combine all three dimensions—interoperability, choreographic abstraction, and no-code authoring—for immersive storytelling.

2.5 Process of Refining Search Parameters

Upon initial screening, several works of interest were identified that, while not review articles, provided valuable domain insights. Recognizing the need for systematic reviews to strengthen the theoretical foundation, a refined search was conducted to identify review articles explicitly aligned with the study's objectives.

This involved systematically revising the search parameters to identify review articles that aligned with the research objectives. The process was divided into three primary thematic areas: Virtual Choreographies, Authoring Tools, and Immersive Storytelling.

2.5.1 Virtual Choreographies

Multiple queries were crafted to identify review papers in this domain, focusing on concepts such as behavior patterns, process models, and workflow models. Two main queries were tested:

- **Query 1:** TITLE ("Virtual Choreographies" OR "Behavior patterns" OR "Orchestration" OR "Process models" OR "Scripts" OR "Sets of actions" OR "Spectrum of tactics" OR "Story grammars" OR "Task sequences" OR "Virtual choreography" OR "Virtual puppetry"

OR "Workflow models") AND TITLE ("survey" OR "review") AND (PUBYEAR > 2019) AND (LIMIT-TO (LANGUAGE, "English")) AND (LIMIT-TO (DOCTYPE, "re"))

- **Query 2:** TITLE ("Virtual choreographies" OR "Virtual choreography" OR "choreographies" OR "choreography" OR "virtual representations") AND TITLE ("survey" OR "review") AND (PUBYEAR > 2019) AND (LIMIT-TO (LANGUAGE, "English"))

Despite retrieving a total of 51 results, none of them were deemed relevant to this study.

2.5.2 Authoring Tools

The search for review papers on authoring tools for storytelling employed three queries:

- **Query 1:** TITLE ("interactive") AND TITLE ("story" OR "drama" OR "storytelling" OR "narrative") AND TITLE ("editor" OR "creator" OR "creation" OR "modeling" OR "authoring tool") AND TITLE ("survey" OR "review") AND (PUBYEAR > 2019) AND (LIMIT-TO (LANGUAGE, "English"))
- **Query 2:** TITLE-ABS-KEY("story" OR "storytelling" OR "stories" OR "narrative" OR "drama") AND TITLE-ABS-KEY("authoring tool" OR "editor" OR "tool" OR "creator" OR "creation" OR "creating") AND TITLE-ABS-KEY("no-code" OR "no code" OR "end-user" OR "end user") AND TITLE("survey" OR "review") AND (PUBYEAR > 2019) AND (LIMIT-TO (LANGUAGE, "English"))
- **Query 3:** TITLE ("story" OR "drama" OR "storytelling" OR "narrative") AND TITLE ("editor" OR "creator" OR "creation" OR "modeling" OR "authoring tool") AND TITLE ("survey" OR "review") AND (PUBYEAR > 2019) AND (LIMIT-TO (LANGUAGE, "English"))

The third query retrieved 67 results, of which one was relevant:

C. Y. Quah and K. H. Ng, "A Systematic Literature Review on Digital Storytelling Authoring Tool in Education: January 2010 to January 2020," *International Journal of Human-Computer Interaction*, vol. 38, no. 9, pp. 851–867, 2022, doi: 10.1080/10447318.2021.1972608.

2.5.3 Immersive Storytelling

Queries targeting immersive storytelling and mixed reality environments were also conducted. The final query was: `TITLE ("story" OR "drama" OR "storytelling" OR "narrative") AND TITLE ("mixed reality" OR "augmented reality" OR "enhanced reality" OR "virtual reality") AND TITLE ("survey" OR "review") AND (PUBYEAR > 2019) AND (LIMIT-TO (LANGUAGE, "English"))`

This query retrieved 71 results, three of which were relevant:

- J. Calvert and M. Hume, "Immersing learners in stories: A systematic literature review of educational narratives in virtual reality," *Australasian Journal of Educational Technology*, vol. 38, no. 5, pp. 45–61, 2022, doi: 10.14742/ajet.7032.
- C. Hadjipanayi et al., "Cultivating empathy through narratives in virtual reality: a review," *Personal and Ubiquitous Computing*, vol. 28, no. 3–4, pp. 507–519, 2024, doi: 10.1007/s00779-024-01812-w.
- K. Dooley, "Conceptualizing and developing narrative-based virtual reality experiences: A review of disciplinary frameworks and approaches to research," *Journal of Screenwriting*, vol. 14, no. 3, pp. 229–249, 2023, doi: 10.1386/josc_00132_1.

2.5.4 Summary of Results

The refined search yielded four relevant review articles across the thematic areas of authoring tools and immersive storytelling. These works provide comprehensive insights into existing tools, frameworks, and methodologies, contributing significantly to the theoretical grounding of this dissertation.

Chapter 3

State of The Art

3.1 Introduction

The exploration of immersive storytelling, virtual choreographies, and authoring tools for augmented and virtual reality has seen significant advances in recent years. To identify the current state of knowledge, four key review papers were selected, as referred to in the previous chapter. These papers were identified with codes, for simplicity during the analysis as seen in table 3.1.

Code	Review Title
A1	Immersing learners in stories: A systematic literature review of educational narratives in virtual reality (Calvert and Hume [11])
A2	Cultivating empathy through narratives in virtual reality: a review (Hadjipanayi et al. [23])
A3	Conceptualizing and developing narrative-based virtual reality experiences: A review of disciplinary frameworks and approaches to research (Dooley [18])
A4	A Systematic Literature Review on Digital Storytelling Authoring Tool in Education: January 2010 to January 2020 (Quah and Ng [40])

Table 3.1: Relevant Reviews

This section synthesizes the insights of these foundational studies to outline the current state of the art, highlighting trends, methodologies, and gaps that guide the direction of this research. Each paper contributes uniquely to understanding the interplay between technology, narrative design, and user accessibility in immersive environments.

To ensure a comprehensive analysis, excerpts from these review papers were systematically analyzed and categorized across several thematic dimensions. These themes - Immersive Storytelling, No-Code Authoring Tools, Interoperability, and Virtual Choreographies - were identified as critical axes that frame the challenges and opportunities in the field. This process involved reviewing the content of each paper to extract relevant insights that align with the research objectives of this dissertation.

While the four selected review papers formed the backbone of this state of the art, the research process also followed a snowballing strategy to deepen the analysis. After identifying relevant excerpts in the reviews, referenced primary studies or tools mentioned therein were explored in greater detail. If these references themselves provided valuable insights into immersive storytelling practices, no-code authoring environments, or XR interoperability, they were also included in the thematic analysis.

This iterative and exploratory process ensured that the review remained open to emerging patterns and technologies that may not have been central in the initial reviews but proved meaningful in the context of this dissertation's objectives. As a result, the following synthesis reflects both the distilled findings of high-level reviews and key contributions from individual primary sources uncovered through this method.

3.2 Relevant Excerpts

The extracted excerpts were then categorized as belonging to one or more of the identified key themes for this research, identified by a numeric code. This way the categorized excerpts could aid to draw conclusions about the state of the art, namely, recurring trends, identified gaps in research and practical implications for the design of immersive storytelling apps and no-code authoring tools.

In order to accomplish this, each text was examined in order to determine its main idea as simply and abstractly as possible. If the topic hadn't been already identified, it was added to the list.

While this metric doesn't necessarily reflect the overall importance of each document, it provides a useful perspective on how the excerpts are distributed across the sources. The following graph (Figure 3.1) illustrates this distribution, showing not only the relevant extracts but also excerpts mentioning future directions of the field.

3.2.1 Classification of excerpts

A total of 14 themes of relevance were identified, with frequencies as seen in Figure 3.1. It is possible to notice a concentration of excerpts in certain topics, reflecting areas of greater focus within the reviewed documents.

The themes with the highest number of excerpts, principles and features for enabling non-technical users to create immersive content without coding, key considerations for designing immersive experiences in VR and AR, and studies highlighting the positive effects of immersive storytelling compared to traditional methods, demonstrate a strong alignment between the literature reviewed and the goals of this research. These areas provide valuable insights and a solid foundation for the development of the proposed tool.

Conversely, some themes, such as frameworks for evaluating immersive content, had fewer excerpts. Additionally, themes like differences in immersion needs for AR and VR, the definition

Figure 3.1: Relevant excerpts in each document

of interoperability in the context of VR and AR, and the role of standards, frameworks, and APIs in ensuring interoperability were not represented in the selected excerpts. This limited representation may suggest less research attention in these areas, but further investigation is needed to confirm whether this trend reflects the broader state of the field.

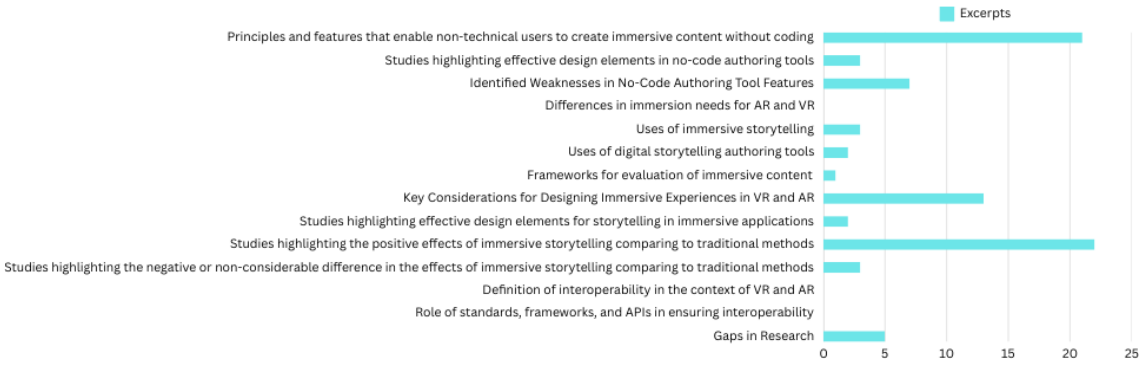

Figure 3.2: Theme frequency of the excerpts

3.3 Key Areas of Research

3.3.1 Immersive Storytelling

Immersive storytelling, a powerful fusion of narrative and technology, offers unprecedented opportunities to engage audiences in virtual and augmented reality. The convergence of virtual reality, interactive design, and narrative frameworks has not only enhanced educational and cultural experiences but also introduced new challenges in crafting meaningful and accessible storytelling environments.

For this axis, it was searched in the documents for the following themes:

- Uses of immersive storytelling
- Key Considerations for Designing Immersive Experiences in VR and AR
- Studies highlighting effective design elements for storytelling in immersive applications
- Studies highlighting the positive effects of immersive storytelling comparing to traditional methods
- Studies highlighting the negative or non-considerable difference in the effects of immersive storytelling comparing to traditional methods
- Frameworks for evaluation of immersive content
- Differences in immersion needs for AR and VR

The analysis of these excerpts led me to conclude that immersive storytelling is predominantly associated with positive effects, as evidenced by 22 excerpts highlighting benefits such as increased engagement, a stronger sense of presence, heightened self-efficacy, and greater user interest (Garneli, Giannakos, and Chorianopoulos [21]). These records suggest that immersive storytelling has significant potential to captivate audiences and enhance educational or cultural experiences, with benefits in knowledge mastery (Abadia, Calvert, and Dasika [2]), knowledge retention (Krokos, Plaisant, and Varshney [28], Meyer, Omdahl, and Makransky [34]), knowledge transfer and task engagement(Bhargava et al. [8]). For example, in a study described by Calvert and Abadia [10], learners scored higher on knowledge tests after experiencing an historical story in VR compared to those who viewed the same experience as desktop 360º video. Some studies also point the benefits in terms of emotional response by the users, for example, Allcoat and Mühlenen [5]

However, there were also three (3) excerpts pointing to studies that found either no significant difference or even negative effects when immersive storytelling was compared to traditional methods (Adams et al. [3], Martey et al. [33]). These instances often occurred in scenarios involving complex instructional elements (Pilegard and Mayer [37]) or when study quality was cited as a limiting factor (Jensen and Konradsen [25]). While these findings are limited, they underscore the importance of carefully designing immersive storytelling experiences to ensure their effectiveness, particularly in educational contexts where cognitive load and clarity are critical.

Additionally, the literature presents several effective design principles and considerations for developers, emphasizing aspects such as user agency, interactive elements, and the integration of spatial storytelling. However, a notable limitation detected is the lack of standardized frameworks for evaluating these experiences, which makes it challenging to compare results across studies or to systematically refine the design and implementation of immersive narratives, being the one proposed by Reyes [42] the only mention to such protocol found during this review. This gap highlights the need for more robust methodologies to assess the impact and quality of immersive storytelling in diverse contexts.

Another important dimension of immersive storytelling is how narrative elements are distributed throughout a virtual space and how user interaction with those elements advances the experience.

In virtual reality, narrative progression is often tied to spatial triggers or object interactions rather than linear sequencing. Rizvic et al. [44] describes a virtual heritage project where users explore a historical environment and activate specific objects to progress through the story. This design allows users to engage with the content in a semi-guided sequence, and as the authors note, “the user can explore the area and play the videos in a precise order, by activating some particular objects placed in the virtual environment. When these objects are activated, the full-screen 360-degree video is played.” The same mechanism is used to allow transitions between alternative realities or reconstructions of the same environment: “they can switch between the underwater environment and the hypothetical reconstruction by activating a particular object placed in the virtual environment”. Rizvic et al. [44] also highlights the importance of directing user attention in

immersive settings, particularly when traditional camera framing is unavailable. Their study observed that "users' attention was directed to the speaking actor, regardless of their viewpoint, and the actor was easily recognized by his coloured appearance. The use of visual cues has particular importance for 360-degree videos that cannot rely on framing and positioning the objects of interest in the center of user's visual attention within the frame." These findings highlight the value of using visual cues, like distinct character design and environmental signals, to guide attention and keep users engaged in non-linear, spatially organized stories.

3.3.2 No-Code Authoring Tools

The rise of no-code authoring tools has democratized digital experience creation, empowering non-technical users—such as educators, historians, and cultural professionals—to design and deploy complex interactive systems without programming expertise. This trend holds particular promise in immersive storytelling, where narrative design often requires domain knowledge more than technical skill.

In reviewing the literature, four recurring themes were identified concerning this axis:

- Principles and features that enable non-technical users to create immersive content without coding;
- Studies highlighting effective interface and design elements;
- Identified limitations and usability challenges in current tools;
- Real-world use cases of digital storytelling platforms.

Design Principles and Enabling Features

A significant portion of the reviewed literature (21 excerpts) focuses on key design principles that lower the entry barrier for non-programmers. Tools like the enhanced concept map by Liu et al. [31] exemplify this, offering a simple model of nodes and links governed by story grammars. Their tool allows users to create episodes, link them logically, and manage story flow at a meta-level.

Design guidelines proposed in multiple sources emphasize interface simplicity and familiarity. Posada and Baranauskas [39] argue that computational complexity should be hidden from users, allowing them to focus on narrative creation. Sadauskas, Byrne, and Atkinson [45] recommends minimalist UI design, advising against overwhelming visual elements such as timelines or exploratory layouts. Consistency is also critical—Pittarello and Bertani [38] suggests using similar icons throughout the process (e.g., from storyboarding to authoring) to ensure clarity and avoid user confusion.

Common Limitations of Existing Tools

Despite these advancements, several usability challenges persist. Seven excerpts describe common shortcomings, including:

- Difficulties accessing and managing multimedia assets like videos and images (Karakoyun and Kuzu [26] and Karakoyun and Yapıcı [27]);
- Challenges in synchronizing narration and multimedia elements (Sheafer [47]);
- The absence of comprehensive platforms that support the full creative pipeline, from ideation to deployment (Quah and Ng [40]).

Addressing these issues is vital to furthering the usability and reach of no-code storytelling environments.

Case Study: Moirai

A compelling example of a modern no-code tool is *Moirai*, described in Torres et al. [49]. Designed for educational serious games, Moirai enables authors to create branching dialogues and learning scenarios without coding, using a single-page web application developed in *Vue.js* and *BaklavaJS*.

Its interface uses color-coded nodes to distinguish narrative functions:

- **Blue:** Start node (only one permitted);
- **Red:** End node (multiple endings allowed);
- **Grey:** Dialogue or decision points;
- **Green:** Game-state modifiers (e.g., setting rooms).

To avoid visual clutter, settings are accessed via buttons that trigger a sidebar (see Figure 3.3).

Moirai also outputs human-readable JSON files, making stories editable both in the tool and externally. Its player, built in Unity and exported to WebGL, supports lightweight execution in browsers by optimizing texture sizes and polygon counts. SCORM compatibility ensures integration with Learning Management Systems (LMS), increasing accessibility.

Testing and Debugging Features

Testing and debugging are central to effective storytelling workflows. *Inform 7*, discussed in Green, Hargood, and Charles [22], includes a feature called *Skein*, which shows a tree of previous test paths to facilitate debugging, though not for editing. *Articy* offers a similar tool—*Journey Mode*—which allows authors to record and replay selected narrative paths for easier regression testing.

Graph-Based Interfaces in Popular Tools

Graph-based interfaces remain a dominant design pattern across no-code platforms. Their visual clarity allows users to understand branching logic at a glance and modify story flow without sifting through raw code. Beyond *Moirai*, notable examples include *Twine*, *Fungus*, *Inform 7*, and *Articy*,

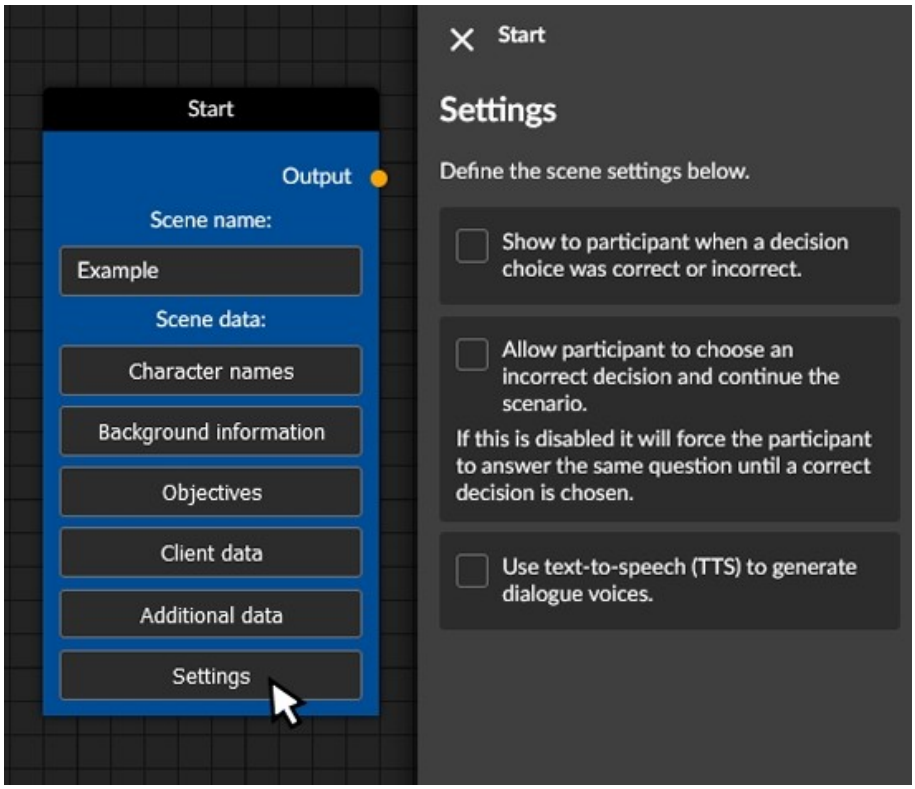

Figure 3.3: Sidebar activated when clicking the "Settings" button of the Start Node in *Moirai*

all highlighted in Green, Hargood, and Charles [22]. These tools reinforce the value of node-link models in authoring complex, non-linear experiences efficiently.

3.3.3 Interoperability

Most existing tools and frameworks focus predominantly on virtual reality (VR), as it offers a higher level of immersion compared to augmented reality (AR), seeming more attractive to researchers and developers seeking to create fully enclosed and highly controlled narrative environments. This emphasis highlights a significant gap in the literature, where the differing requirements for immersion across these two platforms are often overlooked. While VR provides fully enclosed environments, AR overlays digital elements onto the physical world, demanding distinct storytelling strategies. Despite these fundamental differences, current research rarely examines how narratives can be effectively adapted for both domains, leaving educators and developers without clear guidance.

One notable exception is Virtual Veronese by Verhulst et al. [50], which implemented the same narrative and interactive structure across three devices: Oculus Quest (VR), Magic Leap (AR), and Mira Prism (AR). Despite differences in device capabilities and interaction mechanisms, the experience used a unified Unity codebase, identical voiceovers, and consistent stereo video content. This work demonstrated that interoperability is achievable in practical deployments and also revealed key design considerations for adaptive UI per platform.

The Oculus Quest delivered the story within a fully reconstructed virtual chapel, enabling six degrees of freedom (6DOF) movement in a visually isolated environment. In contrast, the Magic Leap version blended the virtual elements with the real-world gallery, also supporting 6DOF but using transparent optics and depth-based persistence tracking to align digital content with physical space. Finally, the Mira Prism offered a more constrained experience, relying on image-based tracking and a semi-reflective visor to project content from an iPhone 8, limited to three degrees of freedom (3DOF) and requiring staff assistance for user interaction. Despite these variations, the story logic and user choices remained the same.

Users rated the VR version higher in terms of presence and enjoyment, although all versions were positively received.

Another notable example of cross-reality design is VRsus guARdian described in Boozayaangool [9], an asymmetric multiplayer game that explicitly explores the interplay between VR and AR platforms. In the experience, one player inhabits a fantasy VR environment as a stealthy infiltrator (see Figure 3.4), while the other uses an AR-enabled mobile device to scan the physical room and act as a guardian searching for intruders (see Figure 3.5). Although the gameplay roles and perspectives differ, the narrative space is shared, with both players affecting the same virtual world from their respective mediums. This design highlights a form of interoperability rooted in narrative and spatial overlap rather than identical user roles. The authors write that "the game utilized each medium's approach towards immersion as a design principle in building a natural, asymmetric play for both players". This aligns with the goal of this thesis to support platform-sensitive storytelling structures that maintain coherent logic across devices, even when the mode of interaction or the narrative frame varies.

Figure 3.4: The inflitrator hides from the AR player, represented by a laser-shooting stone

Figure 3.5: The AR player spots the VR player in the virtual world

3.3.4 Virtual Choreographies

Lacet, Penicheiro, and Morgado [29] point out that one of the fundamental challenges in XR narrative authoring is that interactive digital stories are often ephemeral media. The modes of interaction and the rules that effect them are inextricably bound to a technological platform. Once that platform is rendered obsolete, its stories become unplayable. To address this dual problem of fragility and platform-dependence, the concept of multi-platform virtual choreographies was proposed, combining a high-level semantic description of story logic with platform-specific affordances to preserve and replay interactive narratives across diverse environments.

Virtual choreographies provide a way to describe interactive scenes using a structure that separates what happens in the story from how it is experienced on each platform. The story behavior itself is written using a platform-independent but context-specific language (for example, what characters say or do in a particular setting), while each platform offers its own description of how those behaviors can be carried out based on its technical capabilities. This setup allows a single story to run on different platforms, as long as each one supplies a compatible "platform recipe" that explains how to present the story elements.

Interoperability is achieved by automatically matching the story description with the available actions and features of a platform. Once a story event like "speak to a character" or "go to a location" is defined, it can be reused across many platforms by linking it to platform-specific triggers. The authors exemplify: "Actor 'Darth Vader' (story recipe) can be mapped to Actor 'Black Knight' (castle platform recipe) or to Actor 'Batman' (comics platform recipe). As another example, a verb such as 'Go to' in a story recipe can be associated with 'Move' in a 3D board platform, but with 'Change comic strip background' in a comic strip board platform, since that's how actors 'go' somewhere in a comic strip"

This approach makes it possible for authoring tools to support truly cross-platform storytelling. Writers and educators can create one version of a story and deploy it in both AR and VR without

needing to reprogram it for each medium. Virtual choreographies simplify this process by focusing on high-level actions and meanings rather than device-specific programming.

Even though this model shows great potential, it is still relatively new and lacks a strong base of research or standardized tools. More work is needed to fully explore and refine how virtual choreographies can help connect different platforms and storytelling methods. Doing so could significantly improve how immersive stories are created, shared, and experienced across technologies, especially for non-technical users and interdisciplinary teams.

3.4 Market Exploration

In addition to academic literature, examining commercial authoring tools provides valuable insights into current capabilities, trends, and gaps in the market. This review complements the theoretical research by analyzing real-world solutions available to non-technical users who wish to create immersive experiences.

Several tools were identified from the literature and through broader online searches. These include well-established platforms like MyWebAR [35], Wonda [51], Adobe Aero [4], and CoSpaces Edu [16], as well as more technically advanced or niche offerings such as ShapesXR [46] and the Magnopus [32] (Connected Spaces Platform).

Interface Approaches and Usability Considerations

One of the most prevalent interface patterns is the graph-based model, where story elements and their logic are represented through nodes and connections. However, this approach often suffers from visual clutter as stories grow in complexity. To address this, tools like Articy [7] introduce features such as collapsible groups that let users visually organize nodes into manageable sections, greatly improving usability and clarity in large projects.

Alternative interface paradigms also exist:

- Block-based systems, such as those used by CoSpaces Edu [16], are designed for younger audiences and emphasize simplicity through snapping visual elements together.
- Dropdown-based interaction builders, as seen in Adobe Aero [4], reveal additional interface elements only when needed. For example, once an action is selected, follow-up options appear for defining targets or conditions (see Figures 3.6 and 3.7).

Asset Integration and Authoring Features

Most tools employ a drag-and-drop paradigm for adding assets to the scene, enabling intuitive interaction even for first-time users. MyWebAR [35], for example, supports a wide range of media formats including 3D models, images, videos, sound effects, and background music.

To support non-technical creators, MyWebAR [35] and Wonda [51] also provide starter templates. These templates offer pre-built structures with placeholders and interactive logic already

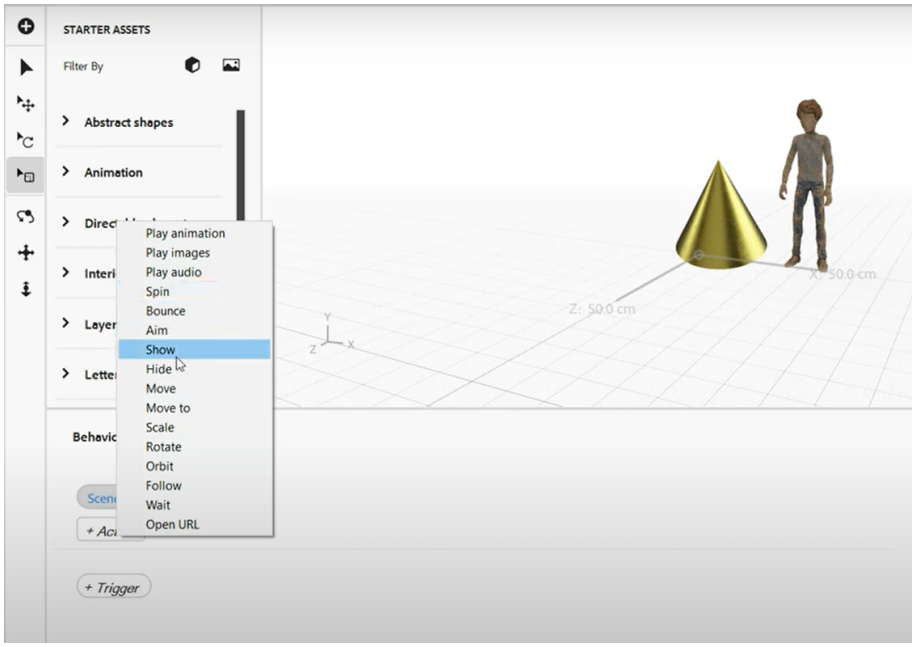

Figure 3.6: Adobe Aero interface: Action selection panel

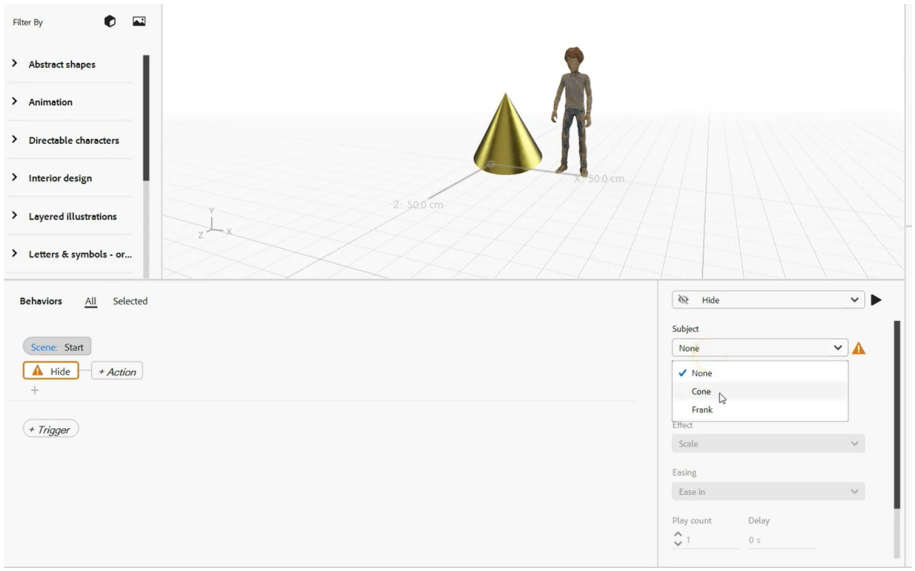

Figure 3.7: Adobe Aero interface: Defining interaction subject

in place, allowing users to focus on content creation without needing to design everything from scratch.

Tools like ShapesXR [46] highlight integration with popular design platforms. Through synchronization with Figma [19], UI components and layouts can be imported and updated in real-time, helping streamline cross-platform workflows between designers and XR developers.

Multi-Platform and Technical Solutions

While most no-code tools focus on ease of use for a single platform, some systems tackle cross-platform challenges at a more technical level. One such system is the Connected Spaces Platform (CSP) by Magnopus [32], originally built for large-scale immersive installations. CSP enables synchronized multi-user experiences across AR, VR, and desktop systems via a shared back-end. Though flexible and compatible with engines like Unity, Unreal, and PlayCanvas, CSP is developer-focused and lacks a visual interface for non-programmers. Its primary strength lies in technical interoperability, not narrative authoring.

Conclusion

Table 3.2: Comparison of Commercial XR Authoring Tools

Tool	**Platform Support**	**Interface Type**	**Narrative Authoring**	**Notable Features / Limitations**
MyWebAR	AR (Web-based)	Drag-and-drop UI	Basic node-linking	Supports media-rich AR scenes, templates available, limited narrative abstraction
Wonda	VR	Template-based UI	Structured but limited logic	Conversational AI simulations, focuses on training use cases
Adobe Aero	AR (iOS, Desktop)	Dropdown menu system	No branching logic	Great asset placement tools, lacks deep interaction modeling
CoSpaces Edu	AR/VR	Block-based scripting	Limited branching logic	Child-friendly, geared toward education, uses block coding
Articy Draft	Game Writing (2D and 3D)	Node graph	Strong branching narrative tools	Powerful for narrative logic, commercial focus, no direct AR/VR export
ShapesXR	VR (Meta Quest)	Spatial UI / Figma Sync	No direct story flow logic	Excellent UI prototyping, no direct authoring of story logic
CSP (Magnopus)	AR/VR/Desktop	Code-based API	No visual authoring	Backend sync across platforms, powerful but developer-focused

This market exploration confirms a growing interest in tools that lower the barrier to immersive content creation. However, most commercial platforms either focus on a single modality (AR or VR), lack narrative abstraction mechanisms like choreographies, or prioritize technical flexibility over no-code usability.

These observations reaffirm the relevance of this dissertation's goals: providing a platform-agnostic, narrative-driven, and accessible authoring solution that empowers creators to work across AR and VR environments using abstract, interoperable representations.

Chapter 4

Development

This chapter presents the development process of the proposed authoring tool, following the Design Science Research Methodology (DSRM) described in Chapter 1. In alignment with the DSRM design cycle, the project was conducted through successive iterations, each refining the artifact to better meet the research objectives.

The work developed in this dissertation represents the third major iteration in an ongoing research line focused on authoring tools for immersive storytelling. The first iteration was introduced by Cardoso [12], who developed a Unity-based 3D editor and player for cultural narratives, incorporating early notions of virtual choreographies. This was followed by the second iteration by Freitas [20], who redesigned the tool as a web-based application with an accessible no-code editor and augmented reality support, setting the stage for broader platform interoperability.

This third iteration expands the tool's capabilities toward full cross-platform storytelling by introducing key architectural changes and virtual choreography abstractions, culminating in a more flexible, modular design.

Three major design cycles were conducted over the course of the project:

- Analysis of the **second iteration** - the prototype developed by Freitas [20], which provided foundational features for AR storytelling and established the concept of node-based narrative authoring.
- The **first phase of the third iteration** extended the tool to support both AR and VR platforms. While functional, this version revealed conceptual and usability issues, especially regarding scalability and redundancy.
- The **second phase of this third iteration** restructured the system around a clear abstraction model based on Virtual Choreographies. This version introduced platform-specific manifests, a central narrative choreography, and a clean separation between story intent and device-specific implementation.

The rest of this chapter details each of these iterations, starting with an analysis of the previous prototype, followed by the design and implementation of the updated system.

4.1 Previous Iteration Analysis

The development of this project builds upon the foundation laid by a previous iteration by Freitas [20], which focused on creating a user-friendly interactive story editor for augmented reality experiences, the Story Weaver. That initial effort addressed the need for non-technical users, such as historians and educators, to create immersive cultural heritage narratives without requiring programming expertise. This section examines the key features, achievements, and limitations of the previous iteration, providing crucial context for the enhancements and innovations proposed in this thesis.

4.1.1 Architecture and implementation

The previous iteration of the project followed a three-component structure: a backend and two frontends — the Story Editor and the Story Player, as seen in figure 4.1. The backend was built using Node.js and Express, handling data management, user information, and assets. The Story Editor and Player frontends were developed with React, ensuring a web-based, responsive environment accessible through browsers.

The Story Editor used libraries like ReactFlow [41] for graph-based visualization of story structures and LeafletJS [30] for interactive maps, while the Player incorporated AR.js [6] and A-Frame [1] for augmented reality experiences. This modular setup allowed seamless data exchange between the editor and player.

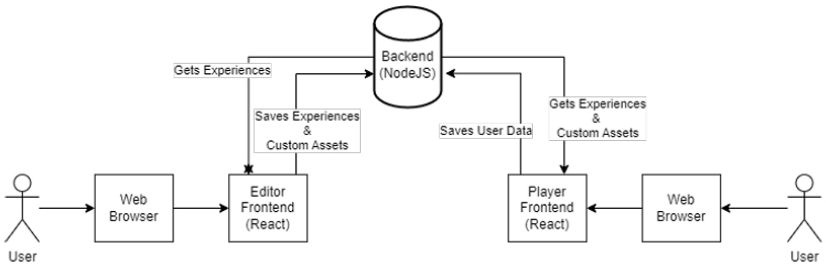

Figure 4.1: Architecture of the tool developed in Freitas [20]

This architecture ensured a balance between flexibility and simplicity, with a strong foundation for future development.

4.1.2 Features and Design choices

Freitas [20] introduced a range of thoughtful design choices aimed at making immersive storytelling accessible to non-technical users.

4.1.2.1 Web-Based Environment:

Both the Story Editor and the Story Player were designed as web-based applications, reducing entry barriers and simplifying access. This approach eliminated the need for complex installations

and ensured the tool could be used on any device with a browser, making it especially suitable for educators and historians without technical expertise.

4.1.2.2 Node-Based Story Creation:

The Story Editor implemented a graph-based, node-and-connection system for building interactive narratives. This visual approach allowed users to construct stories by connecting different types of nodes representing dialogues, decisions, and outcomes, offering a clean and flexible way to represent branching storylines and interactive experiences (see Figure 4.2).

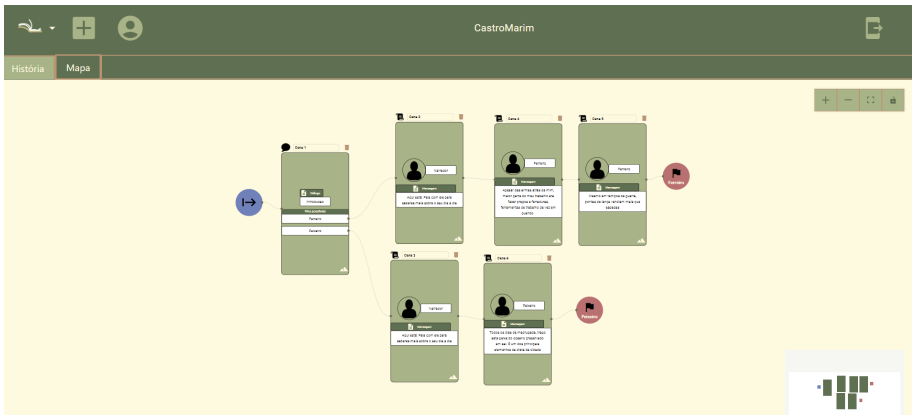

Figure 4.2: Branching storyline in the editor developed in Freitas [20]

Upon selecting a node in the graph, an **inspector panel** would appear on the right side of the screen, allowing users to view and edit the node's properties. This design provided contextual editing without overwhelming the main canvas. Each node type had a unique set of configurable fields tailored to its function. For instance, a dialogue node allowed users to specify the speaking character and the text to be displayed, while a choice node provided options with labeled branches leading to different outcomes. This inspector-based workflow enabled focused interaction, streamlining the authoring process and reducing interface clutter.

4.1.2.3 Tabbed Interface:

To streamline content creation, the Story Editor was divided into three main tabs:

- **Story Tab:** Managed the overall narrative structure, including different scene types like "Begin," "Quiz," and "End."
- **Dialogue Tab:** While visually similar to the Story Tab, this workspace was specialized for creating detailed back-and-forth exchanges using only text and quiz nodes. Each dialogue node in the main storyline links to its own nested dialogue graph that can be edited in this tab, effectively "folding" complex conversations into a separate, focused view. This approach helped prevent visual clutter in the overall story structure by isolating intricate interactions within a contained flow. It allowed authors to design rich branching dialogues—complete

with user decisions and narrative consequences—while keeping the top-level story graph clean and readable.

- **Map Tab:** Enabled authors to create geolocated experiences by placing interactive locations on a visual map. New locations could be defined by clicking on the map interface and assigning coordinates or image positions. Each location was assigned a **type**, such as "Entrance," "Sculpture," or "Artifact", which did not affect behavior but served as a helpful visual indicator to distinguish elements on the map. These tags improved the author's ability to manage spatial logic and gave structure to physical storytelling layouts. (see Figure 4.3).

  This organization made it easier for users to focus on different aspects of story creation without overwhelming them with a single interface.

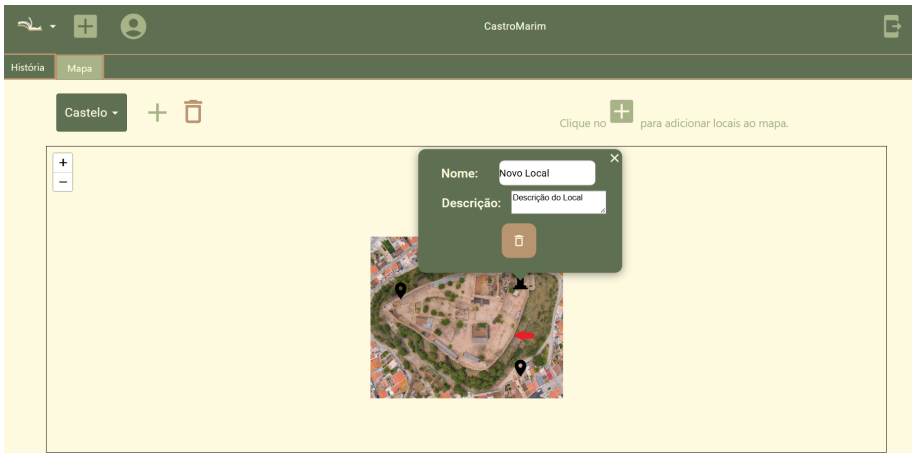

Figure 4.3: Location creation in the Map Tab in Freitas [20]

4.1.2.4 Media Integration

To enhance immersion and expressiveness in interactive narratives, the Story Weaver Editor supported a broad range of media types—including images, videos, audio files, and 3D models. These elements had their own narrative nodes, enabling a richer storytelling experience well suited for educational and cultural heritage contexts. This allowed educators or museum curators to, for example, bring historical artifacts or characters into the user's physical space (see Figure 4.4).

The Story Player, built using `A-Frame` and `AR.js`, rendered this content in-browser on compatible mobile devices. As shown in Figure 4.5, users could interact with the narrative by scanning markers or moving within the geolocated map, causing the associated media to appear in their real environment.

4.1.2.5 Data Structure and Virtual Choreography Foundations

This iteration introduced a flexible and extensible data structure by saving authored experiences in `.json` format: a widely-used and human-readable approach that facilitates further development and integration. Each story file encapsulated essential narrative components such as characters,

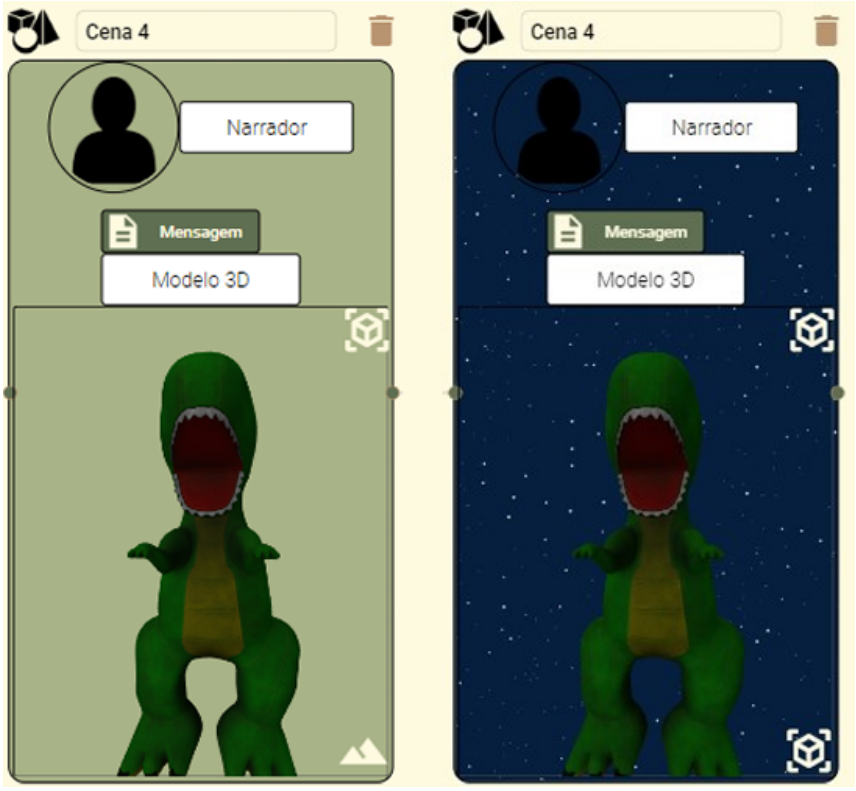

Figure 4.4: Example of a 3D model node in the editor interface from Freitas [20], both with AR disabled and enabled, identifiable by the background

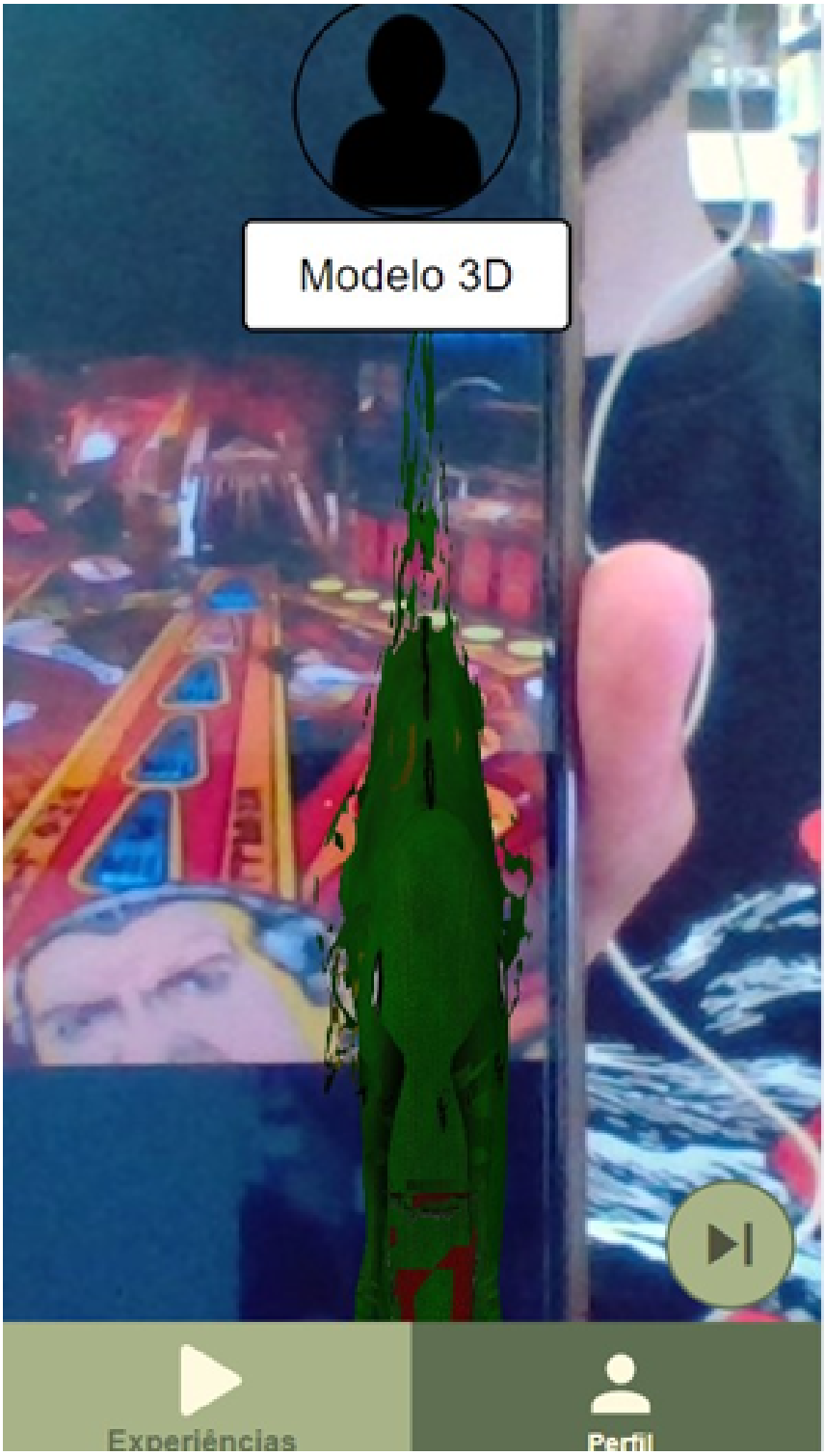

Figure 4.5: Augmented Reality playback in the Player, showing 3D model overlay from Freitas [20]

nodes, maps, edges, and media references. This structured format allowed a clear separation of concerns and offered potential for abstraction, even though platform-specific logic was still partially embedded in the files.

Listing 4.1: Example of JSON structure

```
{
    "id": "93636d71-35f6-482e-aa1d-1cde5aa9cbfa",
    "characters": [
      {
        "id": 0,
        "name": "Andre",
        "description": "O narrador da historia",
        "image": {
          "inputType": "url",
          "filename": "../assets/character_dialogue_node.png",
          "blob": null
        }
      }
    ],
    "description": "O andre precisa de testes e vai pedir ao Miguel. Vamos
        ver o que lhe espera...",
    "edges": [
      {
        "source": "0",
        "sourceHandle": null,
        "target": "f8e27186-cd69-4ff6-815a-41dd8d541177",
        "targetHandle": null,
        "id": "reactflow__edge-0-f8e27186-cd69-4ff6-815a-41dd8d541177"
      }
    ],
    "experienceName": "Andre precisa de testes",
    "lastModified": "2024-06-14T17:15:36.139Z",
    "maps": [
      {
        "id": 0,
        "name": "FEUP",
        "progressionState": "name-given",
        "image": "248e66a7-1a2b-4aaa-a3dc-618b74aeeb6bfejp.png",
        "mapSize": {
          "width": 267,
          "height": 189
        },
        "description": "This is your map's description",
```

```
        "scale": 3.3558146187449744,
        "anchors": [
          {
            "anchorId": 1,
            "anchorType": "anchor",
            "coords": {
              "lat": "41.17841",
              "long": "-8.59497"
            },
            "imgCoords": {
              "x": "121.2130091222467",
              "y": "199.875"
            }
          }
        ]
      }
    ],
    "nodes": [
      {
        "id": "0",
        "position": {
          "x": "-2455.804168669649",
          "y": "-285.9861141108786"
        },
        "type": "beginNode",
        "selected": false,
        "positionAbsolute": {},
        "dragging": false
      }
    ],
    "storyEndings": ["Correu mal", "Correu bem"],
    "tags": [
      { "name": "drama", "color": "#FF0000" },
      { "name": "luta", "color": "#FFA500" }
    ],
    "title": "Miguel e o teste do Andre"
  }
}
```


Although full support for virtual choreographies was not implemented in this version, the JSON-based architecture laid the groundwork for their future integration. The way narrative elements like characters and locations were structured separately from the story graph indicated an early move toward platform-agnostic storytelling. This architectural foundation made it feasible to later introduce higher-level abstractions and the three-manifest approach adopted in this

dissertation.

4.2 Limitations of the Previous Iteration

While the previous iteration, developed by Freitas [20], established a strong foundation for no-code AR storytelling, several limitations became evident during the development of this work. These constraints informed the design choices and architectural changes introduced in the current iteration.

4.2.1 Platform Specificity

The most significant limitation was the focus on augmented reality as the sole deployment platform. Although the editor allowed authors to define story structures and geolocated content, it lacked any form of support or abstraction for virtual reality, limiting its versatility and interoperability. As immersive storytelling increasingly spans multiple platforms, this platform-bound design prevented the reuse of narratives across different environments.

4.2.2 Coupling of Logic and Representation

In the previous system, location-based triggers and platform-specific representations (GPS anchors, AR markers and Image Tracking) were tightly coupled with the narrative structure. There was no separation between the abstract story content and its execution on a particular platform, which hindered flexibility and maintainability. Supporting additional platforms would have required modifying the core story data rather than simply mapping platform behaviors.

4.2.3 Absence of Virtual Choreography Execution

Although the editor introduced elements of virtual choreographies conceptually, such as defining characters and locations, it did not support a formal execution layer for these choreographies. There was no mechanism for interpreting interactions semantically or linking narrative nodes to high-level behaviors. As a result, much of the immersive logic had to be implemented manually and lacked a reusable semantic model.

4.2.4 Lack of VR Rendering Support

Given that the previous iteration focused exclusively on AR experiences, the Player was likewise limited to augmented reality rendering, using technologies such as `A-Frame` and `AR.js`. As the Editor had no provisions for defining 3D environments or VR-specific interactions, the Player did not include infrastructure for importing virtual scenes or enabling spatial navigation in immersive virtual worlds. This constrained the applicability of the system to AR use cases only, leaving virtual reality support as a necessary area for future expansion.

4.2.5 Reactive Storyline

Although the editor supported several node types—such as text, quiz, and 3D model nodes—it offered only limited mechanisms for creating reactive or context-aware storytelling. Some nodes included platform-specific toggles, such as an "Enable AR" checkbox that restricted content visibility to augmented reality contexts, or geolocation-based path nodes defined by start and end coordinates. While these features allowed the narrative to respond to basic spatial conditions, they were hardcoded into specific node types and lacked a unified system for defining arbitrary triggers or reusable conditions. As a result, the tool supported only constrained forms of non-linear progression, making it difficult to author richly interactive or adaptive narratives that respond dynamically to user behavior or environmental context.

These limitations clarified the direction for the next iteration. In particular, they highlighted the need for a formal abstraction of story logic from platform logic, the inclusion of virtual reality support, and the implementation of platform-specific mappings through manifest files. The following sections describe how this research addressed these gaps while trying to preserve the strengths of the original tool.

This project provided an existing and functional codebase that included a web-based graph editor with multiple node types (e.g., text, decisions, and media) for authoring interactive narratives, support for location-based storytelling through GPS coordinates, and integration of marker-based technologies such as QR codes and image tracking. An AR player was also available, allowing authored stories to be experienced in augmented reality. My version of the Story Weaver was conceived and developed on top of this foundation, reusing and extending the original codebase rather than starting from scratch.

4.3 Requirements

This section outlines the requirements for the prototype authoring tool and its corresponding AR and VR playback systems, as intended for development within the scope of this thesis. The requirements are divided into functional requirements, which define the system's core capabilities and features, and non-functional requirements, which describe the system's quality attributes and performance expectations. These requirements were shaped by the insights gained from the exploratory review, the identified research gaps, and the project's goal of enabling immersive, accessible storytelling across platforms.

4.3.1 Functional Requirements

4.3.1.1 Editor Requirements

- **No-Code Development:** The tool must allow users to create interactive and immersive storytelling experiences without requiring programming skills, making it accessible to educators, historians, and other non-technical users.

- **Contextual Help and Tooltips:** To support beginners, the editor must provide informative pop-ups and tooltips that appear when hovering over buttons, icons, and interface elements. These tooltips should briefly explain each feature's function, helping users quickly understand the tool's capabilities without needing extensive documentation.
- **Flexible Story Structure:** The tool must offer a visual representation of story flow, supporting branching narratives and interactive elements based on user choices and actions.
- **Web-Based Environment:** The authoring tool must be accessible through a web browser, eliminating the need for software installation and enabling cross-platform use on different operating systems, minimizing setup effort and making the tool widely accessible.
- **Media Integration:** The tool must support the incorporation of various media types, including images, videos, 3D models, and audio files, to enhance storytelling and immersion.
- **Asset Management:** Users should be able to organize assets into folders or libraries, making it easier to locate, reuse, and manage multimedia elements within projects.
- **Cross-Platform Story Creation:** The editor must ensure that the authored stories can be deployed seamlessly on both AR and VR platforms, maintaining consistent content and interactions across environments.
- **Co-Authoring Support:** The system should allow multiple users to collaborate on the same project, enabling shared editing and collective content creation.
- **Version Control:** Users should be able to track changes made to projects, view previous versions, and roll back to earlier states when needed, ensuring safe and efficient content management.
- **Customizable Templates:** The editor should provide starter templates for common storytelling scenarios to speed up the creation process and offer inspiration, a feature implemented in tools like MyWebAR [35] and Wonda [51].

4.3.1.2 Player Requirements

- **AR and VR Playback:** The player must support the execution of immersive storytelling experiences for AR (mobile/tablet) and VR environments, ensuring each medium provides an optimized experience.
- **Interactive Story Paths:** The player should interpret and execute branching narratives, allowing user choices to shape the progression and outcome of the story.
- **Media Playback:** The player must support smooth and synchronized playback of multimedia content, including images, videos, audio, and 3D models, enhancing immersion without performance interruptions.

- **Virtual Choreography Compliance:** The player application should interpret and execute virtual choreographies, which provide high-level semantic descriptions of interactions, specifying the available actors, their behaviors, and how they combine to ensure interoperability and consistency across immersive experiences.

4.3.1.3 Testing and Deployment Requirements

- **Quick Preview Mode:** The authoring tool should offer an in-editor preview option to rapidly test story flow, interactions, and media without requiring full deployment.
- **Flexible Testing Options:** The tool must allow users to test stories from any specific node or along a predefined path, enabling faster iteration and targeted debugging. Similar features can be found in existing tools, such as the *Skein* feature in *Inform 7*, which visualizes previous test runs with branching paths, and the *Journey Mode* in Articy [7], which allows users to record and replay chosen pathways for easier state restoration and debugging.
- **Platform-Specific Deployment:** The system should support exporting the created experiences in formats compatible with AR and VR platforms, ensuring proper functionality and performance on each medium.
- **Seamless Integration with Player:** Exported content must be easily accessible and fully compatible with the AR and VR players, minimizing configuration and setup effort.

4.3.2 Non-Functional Requirements

The non-functional requirements define the quality attributes and performance expectations of the authoring tool and its associated AR and VR players. These requirements ensure that the system is not only functional but also efficient, reliable, and user-friendly.

- **Usability:** The tool must provide an intuitive and accessible interface, enabling non-technical users to design immersive stories with minimal training and effort.
- **Interoperability:** The system must support seamless deployment of created stories across both AR and VR environments.
- **Scalability:** The tool must be able to handle projects of varying sizes, from simple storylines to large-scale, complex narratives with numerous assets, interactions, and branching paths without performance degradation.
- **Performance Efficiency:** The authoring tool and player must be optimized to minimize loading times, ensure smooth media playback, and maintain high frame rates, even in content-heavy scenarios.
- **Reliability and Stability:** The system must prevent data loss by offering features like autosave, version control, and crash recovery, ensuring users' work is consistently protected.

- **Maintainability:** The system must be designed with a modular architecture, making it easy to update, expand, and maintain over time with minimal impact on existing functionality.
- **Collaboration Support:** The tool must enable co-authoring, allowing multiple users to work on the same project without conflicts or data loss.
- **Cross-Platform Availability:** The authoring tool should be web-based, eliminating installation barriers and providing access from various operating systems and devices through a standard web browser.

4.4 Initial Prototype

4.4.1 Proposed Theory

4.4.1.1 Node-Based Story Model and Triggering Mechanism

Building upon the narrative structure introduced by Freitas [20], this project continues to use a **node-based system** to represent immersive storytelling. Each story is modeled as a directed graph, where nodes represent narrative events (e.g., dialogues, choices, or endings), and edges define the flow between them.

To enhance immersion and interactivity, as discussed in the storytelling section 3.3.1 of the State of the Art, some nodes are designed to be **triggered by user actions**. This means a given part of the story will only unfold once the user performs a specific interaction such as scanning a QR code, walking to a location, or pressing a button on a VR controller, thereby reinforcing spatial storytelling and user agency.

4.4.1.2 Dual-Platform Trigger Model

Following the evaluation of the prototype by Freitas [20], an early design iteration introduced support for both AR and VR platforms. This version proposed a model in which each narrative node could contain **two distinct trigger configurations**: one for Augmented Reality (AR) and one for Virtual Reality (VR). Under this model, a story node could be activated both by scanning a QR code when in AR and by interacting with a 3D object when in VR.

The core idea behind this model was to allow multi-platform deployment without forcing authors to choose a single target medium. However, it relied on a duplicated logic structure, with separate mappings and definitions for each platform embedded directly within the narrative node itself.

This design introduced several conceptual limitations:

- It tightly coupled story logic with platform-specific mechanics, violating the principles of abstraction and reusability.
- Abstract elements like "Entrance" or "Ferreiro" had to be created and referenced separately for AR and VR, leading to data duplication.

- Interactions were defined in low-level terms tied to specific hardware or input methods, rather than abstract user intentions.

4.4.2 Implementation

4.4.2.1 Dual-Platform Trigger Model

In the early prototype, the editor supported platform-specific node triggering by allowing authors to assign distinct activation conditions for both AR and VR. Each narrative node included an expanded inspector interface where users could specify how and when the node should be triggered on each platform individually.

In **AR mode**, triggers were primarily spatial and largely reused the interfaces introduced in the previous iteration. Locations were defined through the 2D map interface using GPS coordinates, as detailed earlier in section 4.1.2.3. These mapped locations could then be assigned as activation conditions for nodes, as shown in Figure 4.7. Additionally, node triggering could also rely on QR codes: by enabling the "Allow AR" option in the node inspector and selecting QR code as the preferred interaction mode, a printable marker would be generated and linked to that specific node (Figure 4.6).

VR spatial mapping was introduced through a newly added interface: the *VR World* tab (Figure 4.9). This interface allowed authors to upload a JSON file containing a list of GameObject names representing actors, locations, and the player's starting point in the scene. The structure followed a basic schema:

Listing 4.2: Example of VR World Configuration File

```
{
    "playerStart": "Entrance",
    "locations": ["Tower", "WeaponStand"],
    "actors": ["Knight", "Peasant", "Blacksmith"]
}
```

To recognize these objects within the Unity scene, the system depended on Unity's built-in tag system. Scene objects had to be tagged accordingly as "Location", "Actor", or "Player" to be detected and included in the export process.

Once the configuration was submitted:

- All actor entries were automatically added to the `Characters` list, named after the 3D object representing them.
- All locations were stored in a special `VR Locations` list, entirely separate from the AR locations defined in the map interface.
- The player's starting position in VR was set to the object named under the playerStart field.

This version lacked linking between AR and VR representations of the same element, authors had to define “Entrance” or “Gate” twice: once on the AR map and once in the VR object list. The resulting structure was functional but fragmented, contributing to the need for an abstraction model introduced later.

Figure 4.6: Node inspector with the “Allow AR” option enabled. This toggle would permit AR-specific triggering, such as associating a QR code with the node.

While this approach enabled minimal VR integration with relatively low effort, it lacked flexibility. Entities in VR could not be explicitly linked to abstract concepts, and no visual feedback or editor-based control was provided. Still, this system established the value of consistent tagging and hinted at the potential for automated scene scanning in immersive storytelling workflows.

More broadly, the architecture in this early version relied on maintaining separate representations of characters, locations, and interactions for each platform. For example, an “Entrance” location had to be created twice: once in the AR map interface with GPS metadata, and again in the VR tab as a virtual object name. Similarly, identifiers for characters and interactions were scoped to the platform in which they were defined, resulting in fragmented logic.

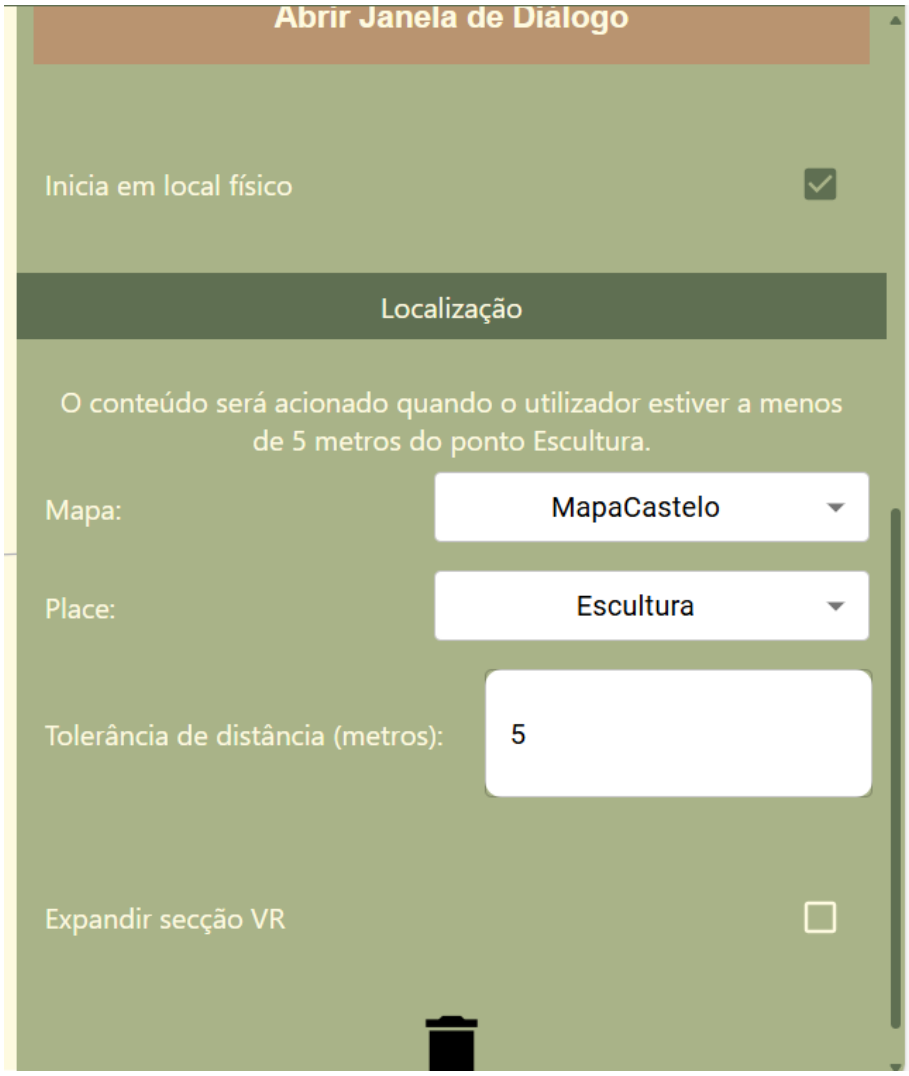

Figure 4.7: Inspector interface for node triggered by GPS location in AR mode

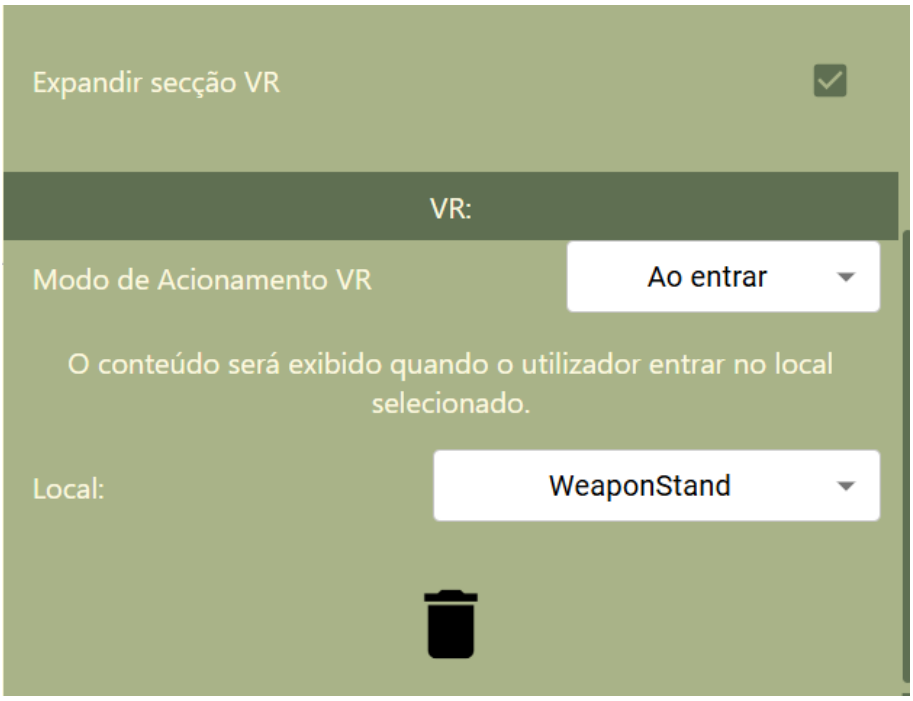

Figure 4.8: Inspector interface for node triggered by reaching a virtual object in VR mode

Figure 4.9: Early version of the VR World tab. Users could upload a list of 3D object names for automatic detection.

This dual-platform design was reflected in the exported JSON structure. Each node could contain embedded blocks describing VR or AR-specific trigger conditions. A representative example is shown below:

Listing 4.3: Example of dual-platform trigger data in early prototype

```
{
"id": "12345",
"action": "text",
"text": "Bem-vindo ao castelo!",
"vr": true,
"vr_type": {
    "trigger_mode": "Ao interagir com ator",
    "actor_id": 1747519147167
},
"ar": true,
    "ar_type": {
    "trigger_mode": "GPS",
    "place": "Coffin"
}
}
```

Once again, while functional, this method introduced several key limitations:

- **Redundant entity definitions:** Abstract characters and locations had to be recreated for each platform.
- **Tight platform coupling:** Narrative logic became entangled with platform-specific data structures.
- **Complex and brittle data:** The JSON output became increasingly verbose and harder to maintain.

Feedback from the thesis advisors confirmed that this dual-binding model contradicted the principles of virtual choreographies and limited extensibility, motivating a shift toward a manifest-based architecture, which offered a cleaner separation of concerns, reduced redundancy, and improved portability across platforms.

4.4.2.2 Preparing the 3D Scene

The 3D scene used for the VR story must be exported in the `.glb` format, which is widely supported across WebXR frameworks. Users developing in Unity must use an external library or plugin to export `.glb` files.

Framework Considerations: For browser-based 3D rendering, the (`.glb`) format was adopted as the standard for scenes, due to its efficiency, portability, and broad support in WebXR frameworks. Several rendering options were evaluated for integration into the Player, with a particular focus on WebXR compatibility, performance, and ease of integration into a web-based pipeline and specially preserving the names of 3D objects in the scene hierarchy, which are essential for correctly linking elements in the choreography system.

Ultimately, **A-Frame** was selected as the rendering framework. A-Frame is a lightweight, declarative VR/AR framework built on top of Three.js, which offers native support for WebXR and can be easily embedded in standard HTML pages. Importantly, prior development work in this project had already integrated A-Frame for earlier AR features, making it the most seamless choice in terms of ecosystem continuity and minimal overhead.

Babylon.js, while not used for rendering in this project, played a valuable role as a Unity exporter. Its Unity Toolkit plugin provided a simple and effective method for exporting scenes to the `.glb` format, preserving mesh hierarchies and structure. However, the system is designed to accept any `.glb` file, meaning users are free to use other export pipelines, including Blender, glTF Exporter for Unity, or other Digital Content Creation tools.

The following table summarizes the evaluated frameworks and their practical trade-offs:

Unity Export Workflow with Babylon Toolkit: Although users may export `.glb` files using any Digital Content Creation tool, this project adopted the **Babylon.js Unity Toolkit** for convenience and compatibility during development.

This toolkit streamlines the process of converting Unity projects into WebXR-ready assets. The following steps summarize the procedure, including encountered issues and their respective resolutions:

1. **Toolkit Installation:** The Babylon Toolkit was retrieved from its official GitHub repository: `https://github.com/BabylonJS/BabylonToolkit/tree/master/Editors/Unity`.

2. **Lighting Configuration Issue:** Upon attempting export, Unity returned an error due to missing lighting settings. This was resolved by navigating to `Window` → `Rendering` → `Lighting`, and creating a new Lighting Settings asset. This configuration is essential as Babylon requires valid lighting data to correctly render materials and scene elements during export.

3. **Scene Export:** Once the toolkit was configured and lighting was set, the scene was exported using the menu path `Tools` → `Babylon` → `Export`. This operation produced a `.glb` file containing all relevant geometry, materials, and scene metadata.

4. **Coordinate System Adjustment:** It is important to note that Unity uses a left-handed coordinate system, whereas `glb` files operate in a right-handed system. If developing the scene

Framework	Pros	Cons
A-Frame	• HTML-based declarative syntax; easy to embed in web pages. • Built-in WebXR support for VR and AR. • Lightweight, fast to load, and well-suited for rapid prototyping. • Previously integrated in the project, ensuring compatibility.	• Lower rendering fidelity than full engines. • Less fine-grained control compared to Three.js or Babylon.js.
Three.js	• Powerful and flexible general-purpose 3D library. • Supports glTF and many other formats. • Extensive documentation and community support.	• Requires manual setup of camera, lighting, controls, and render loop via JavaScript. • No declarative HTML layer—everything must be written in code. • Requires external libraries or plugins to support WebXR features.
Babylon.js	• Rich feature set for games and simulations (physics, LOD, etc.). • High-performance WebXR engine. • Robust tooling for glTF import/export.	• Larger framework size. • Requires full JavaScript scene construction, including asset loading, material setup, and event handling. • More complex than needed for lightweight storytelling use cases.
Unity WebGL	• Enterprise-level graphics and physics fidelity. • Excellent scene tagging and editor tools.	• Very large build sizes; long load times. • Complex to embed into standard web workflows. • WebXR support is limited and not streamlined.

Table 4.1: Comparison of 3D/WebXR frameworks for browser-based `.glb` scene rendering. A-Frame was selected due to its WebXR support, ease of use, and continuity with prior project work.

in Unity, to ensure correct rendering and prevent visual artifacts, the geometry must be converted by flipping the normals and triangle winding order. This can be done within Unity, manually or for example, using a script that inverts normals and swaps triangle indices for each mesh before exporting to `glb` (similar to the one shown in Appendix E.2 or externally using 3D modeling tools like Blender .

A helpful resource for checking exported `.glb` files is available at: `https://gltf-viewer.donmccurdy.com/`

4.4.2.3 Playback Experience and Interaction Flow

The Player application offers two modes of operation: **AR** and **VR**. In both cases, immersive stories can be loaded either by selecting them from the backend, when connected to the online database, or by manually uploading the JSON file containing all the necessary story data. For VR playback, an additional step is required: the user must upload the 3D scene file (in `.glb` format) representing the virtual world after selecting the story.

In **AR mode**, the experience is spatially anchored through GPS-based triggers and visual markers. As users move through the physical environment, the system activates nodes based on proximity or image tracking. The device's camera is used to detect QR codes or tracked images, enabling context-aware progression.

In **VR mode**, the player begins at the position defined in the `playerStart` field of the story, or defaults to the origin if no start point is set. Narrative content—such as text and choices—is rendered as floating panels positioned directly in front of the user for readability. When a node requires a trigger (e.g., interacting with a character or reaching a location), a text cue appears in-world using the format `[interaction] with [target] to continue`. At the same time, the corresponding object is visually marked with a rotating green octahedron (see Figure 4.10) to guide user attention.

This visual cue provides an intuitive and non-intrusive guide for user interaction, in line with best practices described by Rizvic et al. [44] and cited in the Immersive Storytelling section of the State of the Art. As discussed in this section, visual signals and spatial markers are effective tools for guiding attention in non-linear, exploratory narratives. The use of animated geometry as an interaction indicator supports immersion while preserving narrative clarity.

To ensure usability even with complex 3D models composed of multiple sub-meshes, the system uses a hierarchical matching strategy. When the user clicks an object, the system checks whether the clicked mesh has the expected name corresponding to the target actor. If it does not, it recursively checks the parent objects up the hierarchy. This enables accurate interaction even when an actor is implemented as a group of meshes under a common root, maintaining consistent behavior across diverse 3D assets.

Figure 4.10: A panel indicates the required action while a green visual marker highlights the target.

4.4.2.4 Discussion

The initial prototype demonstrated the feasibility of enabling multi-platform immersive storytelling through a node-based architecture. By supporting both AR and VR triggers within the same node and allowing platform-specific configuration of activation conditions, the system successfully validated core ideas around spatial interaction and cross-platform delivery.

However, as the implementation evolved, several architectural and usability challenges became apparent. The reliance on duplicated representations for each platform, such as defining locations twice, added unnecessary complexity to both the authoring process and data structure. Story logic became tightly entangled with platform-specific mechanics, which undermined the flexibility and portability the system aimed to achieve.

Feedback from the advisors emphasized that the approach in this iteration, though directionally correct, contradicted key principles of virtual choreographies, particularly the separation of behavior definition from execution context. As such, this prototype iteration served as a stepping stone: valuable for exploring interaction modalities and integration pipelines, but ultimately limited in scalability and maintainability.

The next section details how these limitations motivated the development of a new, manifest-based abstraction architecture. This revised design introduced a structured separation of platform-agnostic story logic from platform-specific mappings, significantly improving modularity, reusability, and cross-platform authoring workflows.

4.5 Revised Prototype

4.5.1 Proposed Theory

4.5.1.1 Manifest-Based Abstraction

In order to overcome the limitations of platform-bound design and support true cross-platform authoring, the final iteration adopted a manifest-based architecture centered on the principle of abstraction.

At the core of this architecture is the idea of decoupling what happens in a story from how it happens on each platform. This is achieved by defining platform-independent narrative elements that are then mapped to platform-specific behaviors.

Abstract Concepts

Three core abstractions are introduced:

- **Actors**: Entities that participate in the story. These can be characters (like a blacksmith or merchant) or interactive objects (such as a gate or sculpture). Actors can deliver dialogue, be interacted with, or trigger events.
- **Locations**: Semantic places where story actions happen. These are not tied to any specific coordinates or 3D scene objects until mapped.
- **Interactions**: Abstract definitions of user behavior, such as "approach," "talk to," or "observe." These describe what the story expects the user to do, without defining how that action is recognized.

This abstraction enables the same story to be experienced across AR and VR by mapping these general elements to platform-specific implementations.

Three-File Manifest Structure

To support this design, the system relies on three separate JSON files:

- **Default Manifest**: Stores all abstract definitions—actors, locations, and interactions—without referencing any platform-specific technology. It serves as the shared foundation for all versions of the experience.
- **Platform Manifests**: Contain mappings that translate the abstract elements into actionable platform-specific affordances. While this work includes only AR and VR manifests, the same structure can be extended to support other environments.
  - In the AR Manifest, locations are tied to GPS coordinates or visual markers, and interactions may involve scanning or proximity detection.

  - In the VR Manifest, locations and actors are mapped to 3D objects or Unity scene elements, with interactions triggered via controller inputs or collisions.

- **Choreography File**: Encodes the narrative logic of the experience as a directed graph of nodes and edges. Each node references only abstract actors, locations, and interactions defined in the default manifest. This ensures the choreography remains platform-agnostic.

Figure 4.11 illustrates how these three files would interact at runtime. When the experience is launched, the engine loads the `Choreography` file to control the narrative flow. For each action, character, or location mentioned, it queries the `Default Manifest` to understand its abstract meaning, then consults the appropriate `Platform Manifest` to determine how to represent or trigger that element in AR or VR. During the implementation phase, this theory was updated, resulting in a slightly different structure, presented in the next section (4.5.2.1)

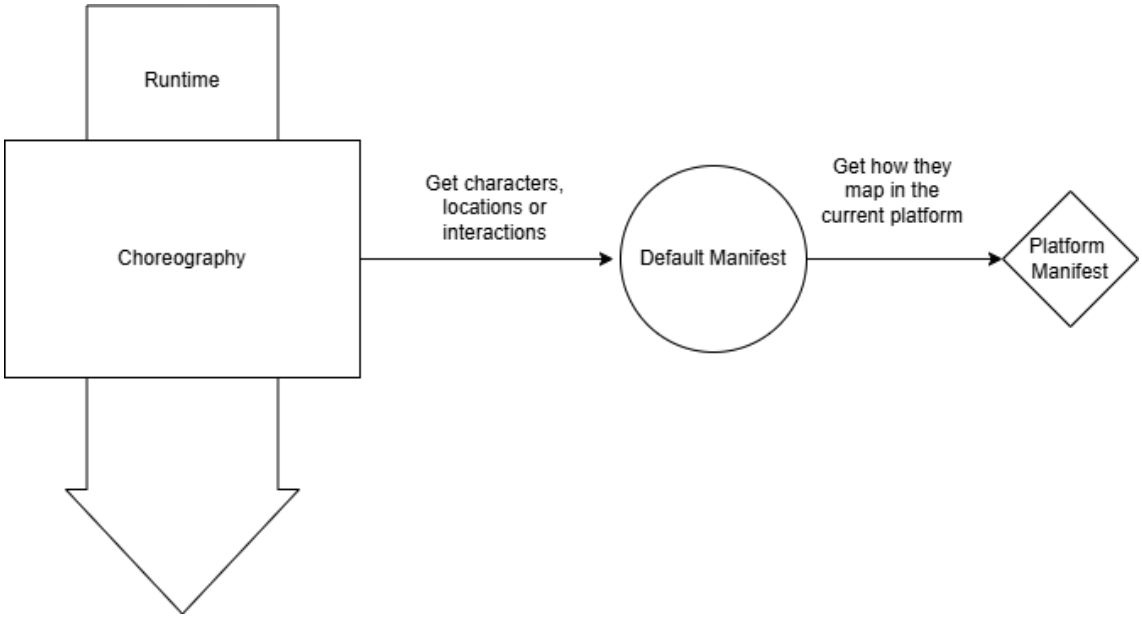

Figure 4.11: Runtime architecture of the manifest-based model. The Choreography defines story logic using abstract elements from the Default Manifest, which are then resolved to platform-specific implementations via the AR/VR Manifest.

This architecture enables:

- **Abstraction** — Narrative logic remains decoupled from platform implementation, "what happens" is completely separated from "how it happens".

- **Reusability** — The same story choreography can be reused on multiple platforms with different mappings.

- **Maintainability** — New platforms require only a new manifest, not a new story design.

This manifest-based abstraction marks a significant improvement over the earlier “dual trigger” model, and aligns closely with the principles of Virtual Choreographies, where behavior is

specified independently from execution context. It also reflects the iterative and problem-driven refinement promoted by the DSR methodology that guides this project.

4.5.1.2 Final Refinement: Combined Choreography Export

A small but meaningful refinement was added to the Player interface: the ability to download a fully **combined choreography file** that merges the three individual inputs - the default Manifest, the platform Manifest, and the choreography — into a single JSON file. Although the system already combined these internally at runtime, this export option was introduced following advisor feedback, as it simplifies distribution, testing, and sharing of authored experiences, demonstrating how iterative design and responsiveness to user feedback can enhance usability even in the late stages of development.

4.5.2 Implementation

4.5.2.1 Choreography Implementation

To support platform-independent authoring, the system separates narrative content from platform-specific implementations using a three-file manifest architecture. These files form the structural backbone for interoperability. Each contains references to the same set of story ingredients, with different mappings depending on the target platform.

1. `The Default Manifest:` This file defines the core story elements shared across all platforms. It includes the story title, project identifier, and lists of characters, locations, and abstract interaction types. Each character or location is identified by a unique ID, which is referenced elsewhere in the system.

Listing 4.4: Excerpt from `defaultManifest.json`

```
{
  "title": "Castro Marim Medieval",
  "characters": [
    {
      "id": 0,
      "name": "Narrador",
      "description": "O narrador da historia",
      "image": { "inputType": "url", "filename": "../assets/
          character_dialogue_node.png" }
    },
    ...
  ],
  "locations": [
    { "id": "22", "name": "Entrada" },
```

```
    { "id": "35", "name": "Banca Ferreiro" }
  ],
  "interactions": [
    { "type": "talk_to", "label": "Falar" },
    { "type": "approach", "label": "Aproximar" }
  ]
}
```

2. `The AR Manifest:` The AR manifest maps abstract story elements to AR-specific interaction methods and spatial references. Characters and locations are linked to QR codes or GPS coordinates, while interaction types are assigned AR-compatible input methods (e.g., `qr_code`, `gps`). The value field for triggers of the QR code type refer to the common name of the generated `.iset`, `.fset` and `.fset3` files.

Listing 4.5: Excerpt from `arManifest.json`

```
{
  "characters": [
    {
      "id": "12",
      "name": "Ferreiro",
      "trigger_type": { "type": "qr", "value": "ferreiro" }
    }
  ],
  "locations": [
    {
      "id": "35",
      "name": "Banca Ferreiro",
      "trigger_type": { "type": "gps", "lat": 20.33, "lng": -20.68 }
    }
  ],
  "interactions": [
    { "type": "talk_to", "label": "Falar", "methodAr": "qr_code" }
  ]
}
```

See also Appendix A (Listing A.1)

3. `The VR Manifest:` The VR manifest maps characters and locations to 3D objects (referenced by name in the Unity scene), and assigns each interaction type a corresponding VR input method (e.g., controller button, proximity detection).

Listing 4.6: Excerpt from `vrManifest.json`

```
{
  "characters": [
    {
      "id": "12",
      "name": "Ferreiro",
      "threeDObject": "blacksmith"
    }
  ],
  "locations": [
    {
      "id": "35",
      "name": "Banca Ferreiro",
      "threeDObject": "WeaponStand"
    }
  ],
  "interactions": [
    { "type": "talk_to", "label": "Falar", "methodVr": "primary" },
    {"type": "approach", "label": "Aproximar", "methodVr": "proximity"}
  ]
}
```

See also Appendixes B and C (Listing B.1 and C.1).

This manifest-based separation allows users to design the world the story takes place once and deploy them seamlessly across platforms. The system can dynamically resolve triggers and scene logic using the appropriate manifest depending on the environment (AR or VR), without altering the underlying narrative structure.

4. `The Story Choreography:` The `Choreography` file defines the dynamic flow of the story through a directed graph structure composed of interconnected narrative nodes. Each node contains an `id`, an `action` type, optional content (`text`, `actor`, etc.), and a `goToStep` reference, pointing to the next node in the sequence. The file also contains basic metadata about the experience, including its title and author.

Nodes are classified by their `action` field.

This field that describes the event or narrative function taking place at that point in the story. Actions are abstract representations of storytelling units, and they define how the system should behave when the node is reached during playback.

Rather than being tied to a fixed set of behaviors, the `action` field is designed to be open-ended and extensible. Developers may define a wide variety of actions—from dialogue lines and choices to animations, audio cues, mini-games, or external service calls—depending on the platform's capabilities and the authoring needs.

In this way, the choreography format supports future growth: new types of interactions or content can be integrated into the narrative without changing the underlying structure of the system. While this project currently implements a focused subset of actions, the architecture is designed to accommodate far more complex or multimedia-rich experiences going forward.

Each node may also include an optional `trigger` block, which links a specific interaction to a character or location. This enables situational activation: a node only becomes active when a user performs the specified action (e.g., "talk_to") with the target entity (e.g., "Ferreiro").

The following examples illustrate core node types and their interactions:

Listing 4.7: Begin node example

```
{
  "id": "0",
  "action": "begin",
  "location": "Entrada",
  "goToStep": "a73bdbe7-3bda-4ed6-84f7-ed1e7d977338"
}
```

Listing 4.8: Text node with trigger

```
{
  "id": "24617b23-26e2-45d8-a89a-8075b4804311",
  "action": "text",
  "actor": {
    "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
    "name": "Ferreiro"
  },
  "trigger": {
    "interaction": "talk_to",
    "target": "Ferreiro"
  },
  "data": {
    "text": "Apesar das armas atras de mim, maior parte do meu trabalho
        era fazer pregos e ferraduras..."
  },
  "goToStep": "a81f333e-7bfb-4fc4-b49c-d5a2b1da44f6"
}
```

Listing 4.9: Choice node with branching

```
{
  "id": "5feb3013-35a4-435f-87b5-42efd2a429ef",
  "action": "choice",
  "actor": {
    "id": 0,
    "name": "Narrador"
  },
  "data": {
    "text": "Que banca gostarias de visitar",
    "options": [
      { "label": "Ferreiro", "goToStep": "2" },
      { "label": "Peixeiro", "goToStep": "fd8a5668-089b-46c8-ae68-49c8981d
          4b77" }
    ]
  }
}
```

For a more extensive example, see Appendix D (Listing D.1).

Runtime Merging of Manifests: While the manifest structure was conceptually designed to operate as three independent files, practical implementation revealed that merging them during the story-loading phase provided significant runtime benefits. At launch, the system parses all three files and merges their contents into a unified in-memory representation. This merged file includes a new field, *platformType*, which explicitly identifies the target environment ("VR" or "AR"), allowing the Player engine to adapt its behavior accordingly. While nodes in the story still reference abstract IDs for characters, locations, and interactions, the merged structure centralizes all relevant mappings and metadata, such as 3D object names, GPS coordinates, and input methods. This design both preserves the logical separation of concerns and streamlines runtime evaluation by eliminating the need to reference multiple files during playback.

Listing 4.10: Excerpt from Merged Manifest File (VR)

```
{
    "characters": [
        {
            "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
            "name": "Ferreiro",
            "threeDObject": "blacksmith"
        }
    ],
    "interactions": [
        {
            "type": "talk_to",
```

```
            "label": "Falar",
            "methodVr": "primary"
            }
    ],
    "story": [
        {
            "id": "24617b23-26e2-45d8-a89a-8075b4804311",
            "action": "text",
            "actor": {
                "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
                "name": "Ferreiro"
            },
            "trigger": {
                "interaction": "talk_to",
                "target": "Ferreiro"
            },
            "data": {
                "text": "Apesar das armas atras de mim, maior parte do meu
                    trabalho era fazer pregos e ferraduras..."
            }
        }
    ]
}
```

This fusion of manifests allows the Player engine to operate without cross-referencing multiple files during execution, enabling a more streamlined playback loop while preserving the modularity and reusability of the architecture.

4.5.2.2 Creating the Narrative

This section outlines the intended authoring workflow for constructing immersive narratives using the proposed tool.

Before authoring the narrative flow, users must define the immersive environment by creating locations, characters, and their mappings in both AR and VR contexts. The following steps outline the preparation workflow within the Editor:

1. **Define Abstract Elements**: Users begin by defining the building blocks of the story world—locations, characters, and interaction types. Each of these elements is modeled abstractly, ensuring reusability and platform independence. These blocks can be created and edited opening the respective window in the top bar of the tool (see Figure 4.12)

    - The **Characters window** provides a space to define the cast of the experience. Each entry includes a character name and may be accompanied by a representative image

or short description to aid visual identification and narrative clarity (see Figures 4.13 and 4.14).

- Within the **Locations window**, authors specify the key places that structure the story world. Locations are identified by name and can be optionally assigned a semantic type (e.g., Artifact, Entrance), which currently influences iconography in the editor, but may later support logic-based filtering or conditional triggers (see Figure 4.15).
- The Interactions window serves as the configuration point for abstract user actions such as "Speak To," "Inspect," "Approach"... Each action can be mapped to platform-specific input methods for both AR or VR environments (see Figure 4.16).

  Available input types include:

  - **AR Inputs**
    - GPS coordinates
    - QR code scanning
    - Image tracking
  - **VR Inputs**
    - Proximity to 3D object
    - Primary button
    - Secondary button

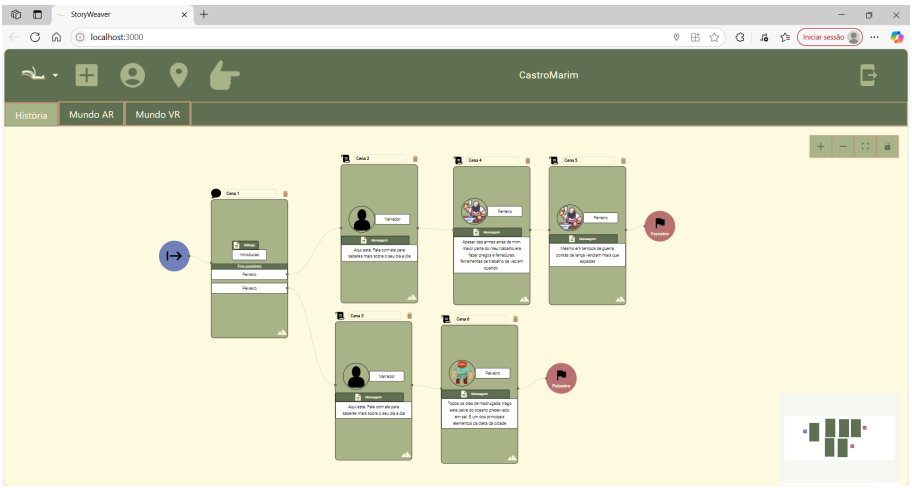

Figure 4.12: Main Window

2. **VR World Configuration** The new *VR World* tab provides a graphical interface for linking abstract story elements—such as characters and locations—to specific 3D objects within a scene (see Figure 4.17).

   - The user can upload a simple JSON file listing the scene objects intended to serve as locations, characters or interactable artifacts. To create this file, the user can either

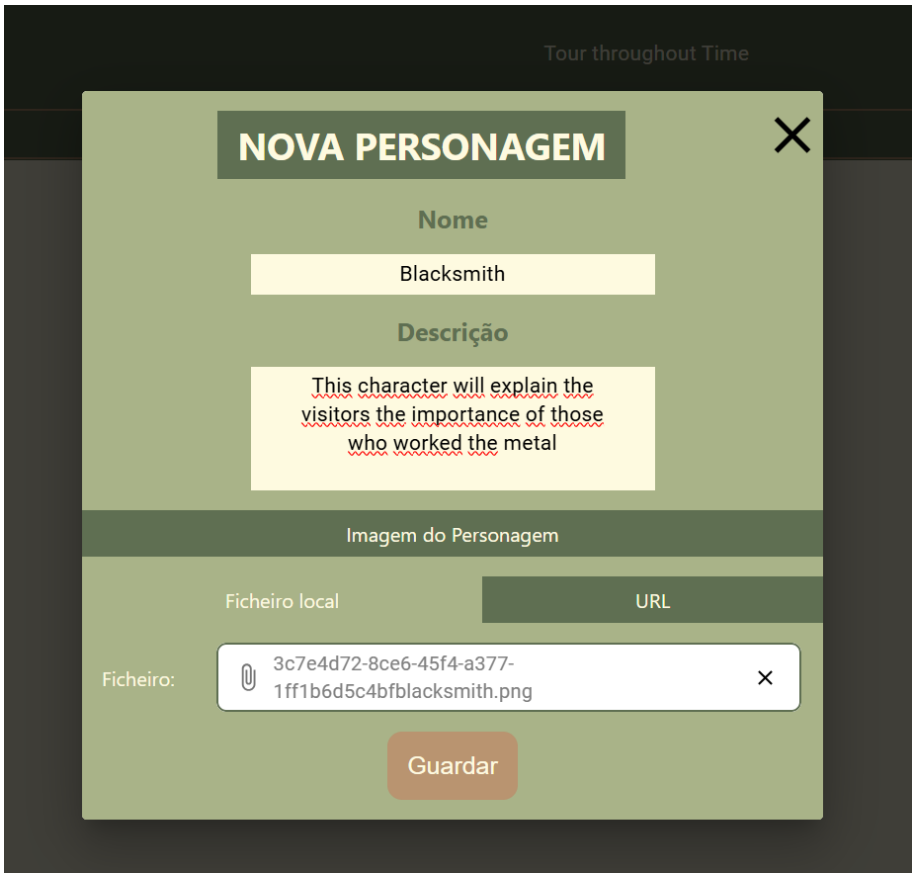

Figure 4.13: Character creation window where users define names, descriptions, and optional images.

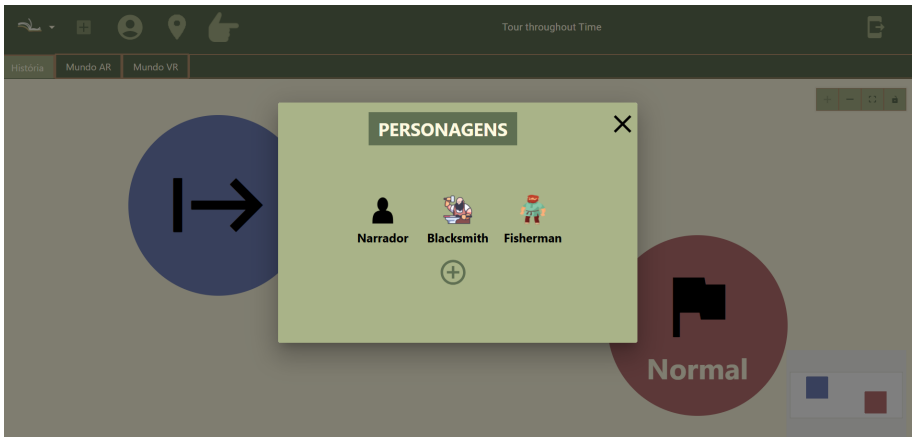

Figure 4.14: Overview of created characters, showing each entry and its associated visual representation.

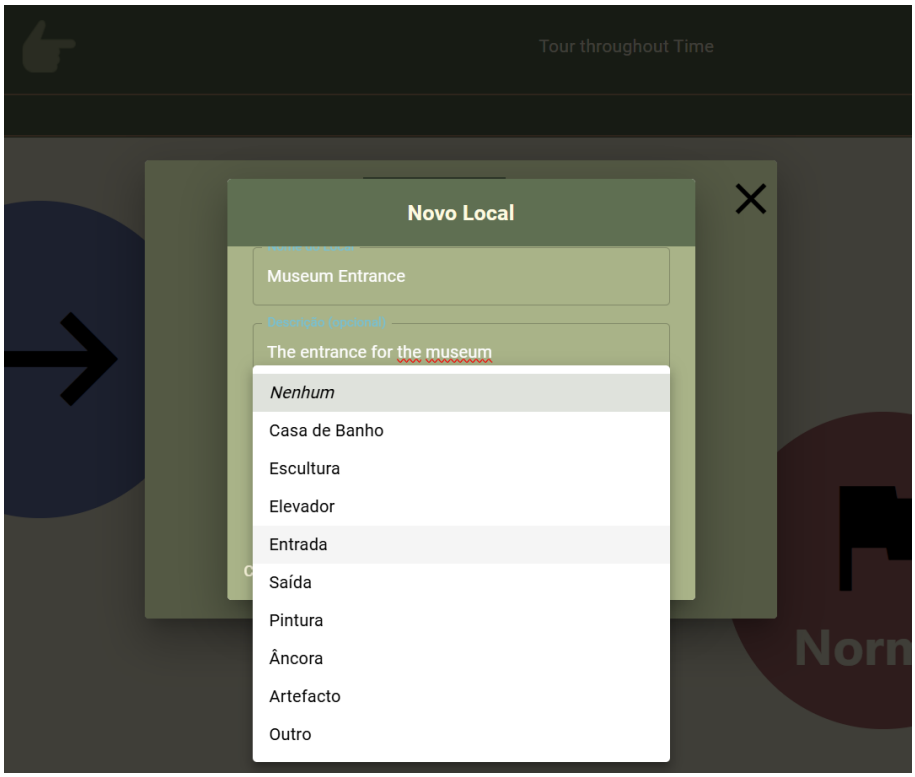

Figure 4.15: Location creation popup with the type dropdown open.

Figure 4.16: Interaction popup: showing a new interaction in the process of being created, alongside previously defined ones.

click “Download Unity Helper” to retrieve a Unity script that automates JSON generation by scanning for GameObjects in the scene tagged with "Location" or "Actor" (see Appendix E.1) or by simply create it by hand. If so, the user should ensure the file adheres to the format:

```
{
  "locations": ["FishStand", "WeaponStand", "StartPosition"],
  "actors":    ["blacksmith", "Fisherman"]
}
```

This format replaces the previous iteration’s use of a dedicated `playerStart` field. In the updated system, authors define the player’s initial position directly through the `Begin Node` in the choreography, selecting any location defined in the manifest. This shift simplifies the structure and grants authors greater narrative control over where the experience begins, decoupling spatial setup from hardcoded scene references.

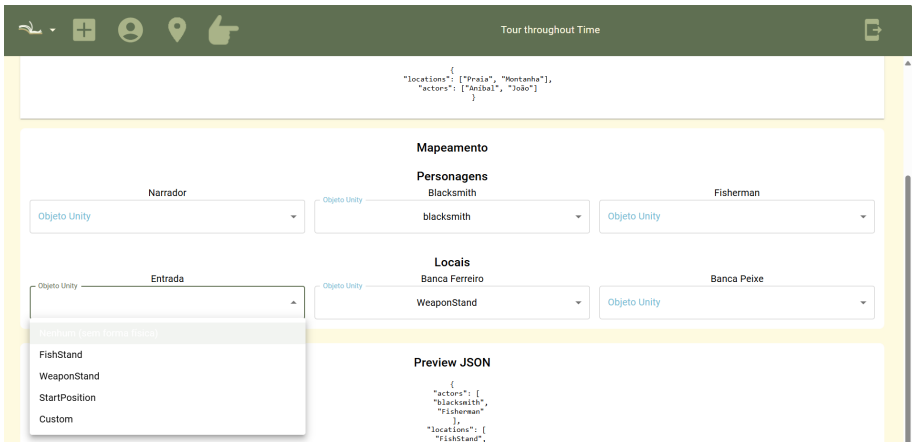

Figure 4.17: VR World mapping interface, where users associate abstract characters and locations with 3D objects in the virtual world.

3. **AR World Configuration** The *AR World* graphical interface has two sub-interfaces: the *Map* sub-tab and the *QR/Image Tracking* sub-tab.

   - The **Map Tab** allows the users to place locations geographically in the physical world:
     - The process begins with uploading a 2D image of the physical environment, such as a floor plan, site diagram, or aerial photograph. An example using an overhead view of a castle is shown in Figure 4.18.
     - To calibrate the coordinate system, two anchor points must be defined, ideally placed in opposite corners of the image. These points are set by clicking on the image and specifying either their real-world GPS coordinates or using the “Use my position” feature to capture the current location. These anchors provide a

reference framework for interpolating the positions of all other mapped locations (Figure 4.18).

  - With the anchor points established, abstract locations can then be dragged onto their intended positions on the map. The system calculates and assigns precise GPS coordinates to each location based on their relative positions to the anchors (Figure 4.19). When selected, a location reveals its computed coordinates and other metadata (Figure 4.20).

- The **QR/Image Tracking Tab** provides a mechanism for linking abstract characters and locations to visual markers that can be used in augmented reality experiences.
  - Users begin by selecting an entity, either a character or a location, from a drop-down menu, as illustrated in Figure 4.21.
  - Once selected, the entity can be associated with either a system-generated QR code or a user-provided image. The "Generate QR Code" option creates a printable marker that can be physically placed in the environment, while "Upload Image" allows custom visual assets to be used as image tracking targets.
  - The interface provides immediate visual feedback on the linking status of each entity: those with associated markers are displayed with a green checkmark, while unlinked entities are marked with a red cross (Figure 4.22).

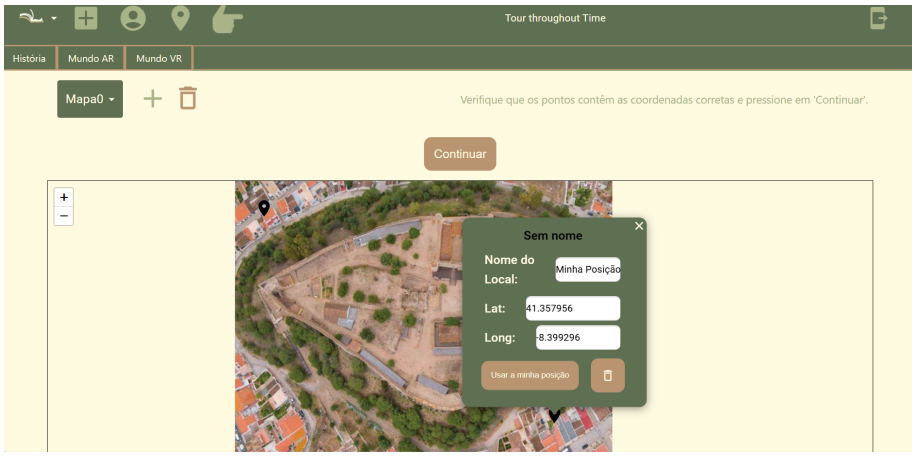

Figure 4.18: Map interface with a castle aerial image. The user is placing the first of two anchor points used to calibrate the GPS projection.

Once these steps are complete, the Editor has all required spatial and semantic mappings to bind the abstract story elements to both AR and VR environments. Users may now proceed to construct the choreography graph and export the manifest files.

The graph-based interface of the editor allows the user to visually compose the storyline through interconnected nodes, each representing a narrative element or decision point.

When a new project is created, the graph begins with two predefined nodes: a **Begin Node** and an **End Node**.

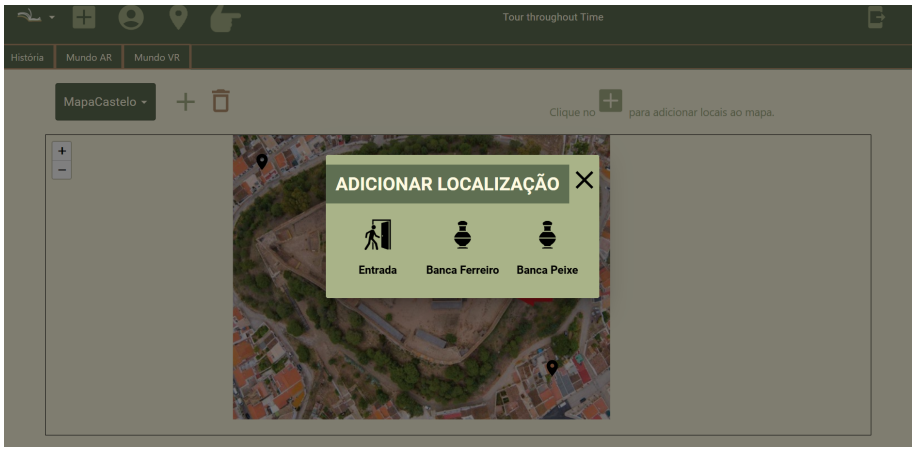

Figure 4.19: Popup for adding a location to the AR map. The user selects which abstract location to place on the image, triggering automatic GPS interpolation.

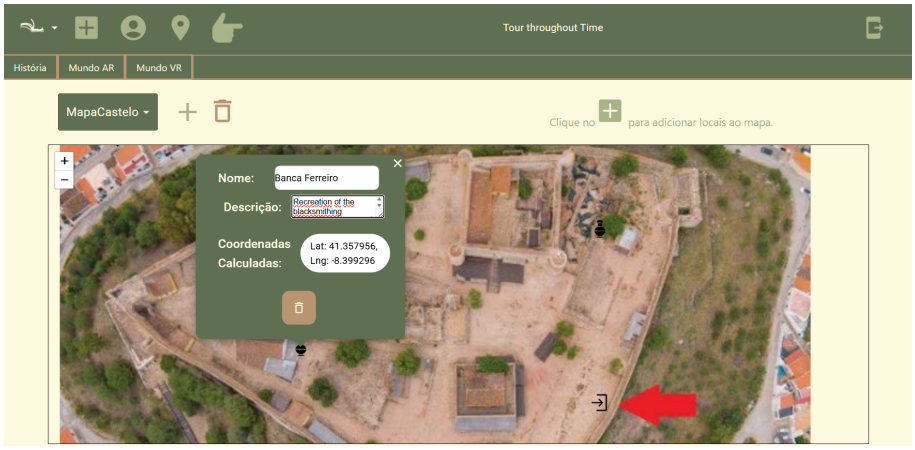

Figure 4.20: AR map with three locations placed. The selected location displays its details and calculated coordinates based on the anchor reference system.

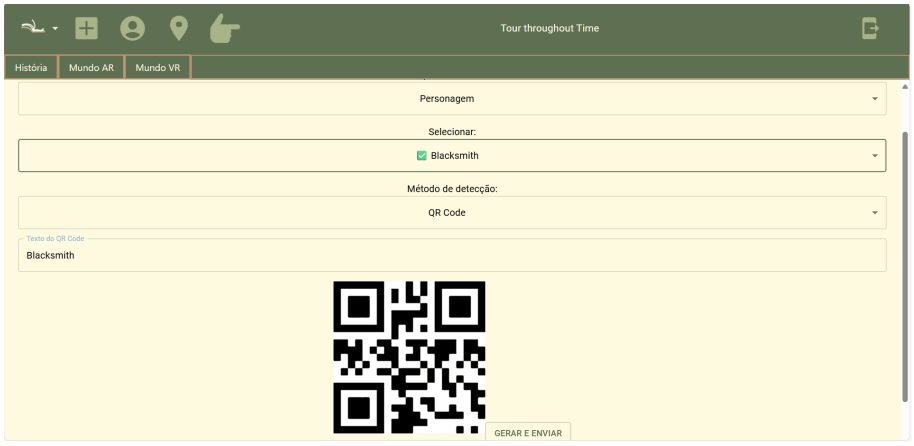

Figure 4.21: Overview of the QR/Image Tracking interface.

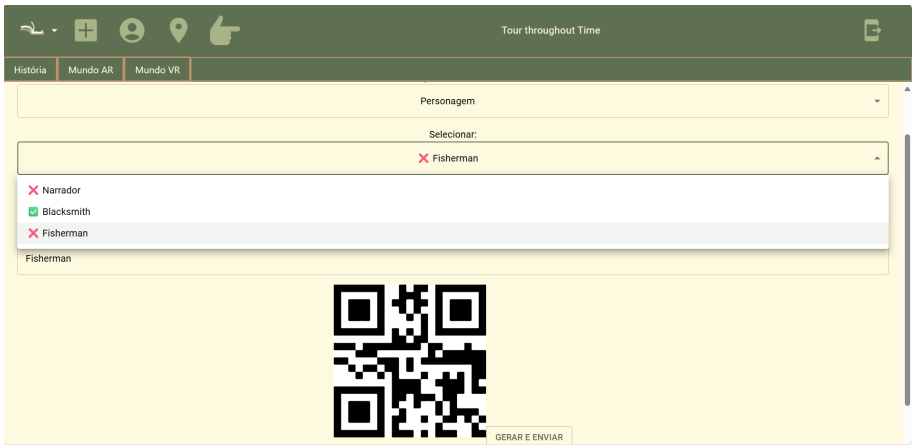

Figure 4.22: QR code generation window with the character dropdown open. Characters with assigned codes are marked with a green check, and those without are marked with a red cross.

- The **Begin Node** is unique and mandatory. It includes a single configurable property: an optional starting position, which determines the location where the user begins in the VR environment.
- The **End Node** marks the conclusion of a narrative branch. It includes an identifier field that allows for differentiated endings, helping the player understand which outcome was achieved.

This application builds upon the foundation laid by the work of Freitas [20], which introduced a wide variety of node types for narrative construction. In this system, nodes support multiple incoming connections but typically only one outgoing edge, with a few exceptions. However, during the development of this project, only a core subset of these node types was fully restructured to comply with the new abstraction model. These three adapted node types—Text, Quiz, and Dialogue—form the foundation of the current system and were prioritized due to their versatility and relevance in storytelling.

- **Text Node**: Represents a single excerpt of dialogue or narration. It includes two fields: one for selecting the character delivering the message, and another for the message text itself.
- **Quiz Node**: A branching decision point presented to the user. It includes a character (who poses the question), a text field for the question itself, and a dynamic list of answer options. Each option corresponds to a separate output edge, allowing the story to branch into multiple distinct paths based on the user's choice.
- **Dialogue Node**: A special nested container node used to group together multiple Text and Quiz Nodes. This is particularly useful for organizing character conversations. Internally, the Dialogue Node includes its own subgraph, and may have multiple output edges depending on the structure and outcomes defined within it.

Each of these nodes can optionally be associated with a user-triggered interaction. By enabling the **"Activated by trigger"** checkbox, the user can specify:

- The **interaction** (e.g. “Speak To”, “Enter”)
- The **target type** (Character or Location)
- The specific **target** (e.g. “Darth Vader”, “Castle Gate”)

When this setting is enabled, the node will only be executed during playback once the player performs the specified interaction with the defined target. This feature enables context-sensitive storytelling and supports both exploratory and goal-based narrative structures within immersive environments.

4.5.2.3 Player Interface

The Player is a web-based application designed to load and execute immersive narrative experiences in either AR or VR. Its interface provides a streamlined method for users to test and view their authored stories, whether built from separate manifest files or pre-compiled as a single file.

The Player interface (Figure 4.23) is divided into two main sections, offering two different workflows for launching an immersive experience:

- **Three-File Input Mode:** This is the primary method for loading modular experiences. It includes:
  - A **platform toggle button** to switch between **AR** and **VR** modes, ensuring that the appropriate platform manifest is loaded.
  - Three file input fields:
    - `The default Manifest` – defines the core characters, locations, and interactions.
    - `Platform-specific manifest` – either `AR` or `VR`, depending on the selected mode.
    - `The Choreography` – defines the story flow.

  After all files are selected, the user clicks `Start` and, before execution begins, a pop-up window prompts the user to choose between:

  1. **Proceeding normally** with the uploaded files, or
  2. **Downloading a combined choreography** file, a unified `.json` that merges all the loaded data into a single exportable file.

- **Combined Choreography Mode:** This alternative section allows users to load the single, pre-assembled `.json` file mentioned earlier, containing all necessary data (default manifest, platform mappings, and choreography). This method streamlines testing and distribution, as allows the users to restart stories immediately.

If the user selects **VR mode**, an additional prompt appears after loading the choreography, asking the user to upload the corresponding `.glb` file. This 3D model file represents the VR environment and must contain the objects named in the manifest to support character and location binding during runtime.

Figure 4.23: Player interface with two modes of selection

The complete source code for the version of the Story Weaver, both Editor and Player, developed in this dissertation is publicly available at the following repository: Ribeiro [43]

This implementation demonstrates how a unified authoring model can support both AR and VR experiences from a single abstract narrative structure. By separating platform-specific concerns into dedicated manifests and resolving them dynamically at runtime, the system maintains flexibility and consistency across environments. Features such as context-aware triggers, spatial anchoring, and visual guidance cues collectively support immersive, user-driven storytelling. With the foundational components now in place, the next chapter explores the system's practical evaluation and its implications for future development.

Chapter 5

Discussion and Evaluation

This chapter presents a critical discussion of the final prototype developed over the course of this dissertation. It evaluates how effectively the system met the defined objectives and design requirements, identifies limitations and unmet features, and outlines scenarios for future testing and real-world applicability. The discussion reflects on both the architectural foundation established and the practical considerations for usability, storytelling expressiveness, and cross-platform deployment.

5.1 Requirements not met

While this iteration successfully achieved its core technical goals, namely the implementation of a manifest-based architecture, support for both AR and VR platforms, using **virtual choreographies** and a functioning no-code editor and player, some of the more user experience–oriented requirements outlined earlier were not fully realized. These limitations reflect both the prioritization of foundational system design and time constraints during the development process.

5.1.1 Editor Features Not Implemented

- **Contextual Help and Tooltips:** Although some tooltips and visual cues are present in the interface, the system lacks a fully developed contextual help system. More robust in-app guidance, such as onboarding prompts or inline explanations, could improve usability for first-time users.
- **Co-Authoring and Version Control:** Multi-user collaboration and version management were not included in this iteration. These features are crucial for larger teams or educational environments, where collaborative authoring and rollback capabilities improve reliability and productivity.
- **Customizable Templates:** The tool does not currently offer pre-defined templates for common storytelling structures. While authors can build stories from scratch, starter templates would significantly accelerate early-stage content creation, especially for non-technical users.

- **Flexible Testing Tools:** While the Player supports full playback of authored stories, the editor lacks features such as the ability to preview from arbitrary nodes or simulate specific story branches. These functionalities—found in systems like Inform 7's Skein or Articy's Journey Mode—are useful for debugging and iteration but were left out in this version due to focus on core architecture.

5.1.2 Focus on Architectural Foundation

These features were deferred in favor of building a stable and extensible foundation for the system. The architectural work completed in this iteration, such as the separation of concerns via manifests, choreography abstraction, and AR/VR playback, lays the groundwork for future enhancements. As such, usability improvements and collaborative functionality are recommended as high-impact future work.

5.2 Future Test Scenarios

Due to time constraints and prioritization of design and implementation phases, a formal user testing campaign was not conducted during this dissertation. However, plans for evaluation were outlined and can serve as the basis for future testing scenarios, particularly if the tool is extended or deployed in real-world educational or cultural heritage contexts.

The following elements define a structured test plan for future work:

- **Target Audience:** Non-technical users such as educators, museum staff, and historians, who would benefit most from no-code immersive storytelling tools.
- **Scenario Design:** Participants would be asked to create short immersive narratives using predefined content (for example, a widely known story like Red Riding Hood), focusing on placing locations, defining interactions, and linking narrative logic using the node-based editor.
- **Platform Testing:** Stories would be tested in both AR and VR playback modes to validate platform interoperability and the flexibility of the manifest-based architecture.
- **Metrics:**
  - **Usability:** Measured through task completion time, navigation ease, and interface clarity.
  - **Interoperability:** Assessed by verifying that a single choreography can be executed consistently in both AR and VR with minimal reconfiguration.
  - **Narrative Quality:** Subjective feedback on how clearly and effectively users could communicate their story through the tool.

- **Feedback Collection:** Surveys, interviews, and observational notes would be used to identify points of confusion, inefficiencies, or unmet needs. These insights would guide the next iteration of interface and feature improvements.

While informal feedback from the academic advisors played a role in shaping the tool, a formal evaluation phase remains essential to verify its effectiveness with real users and ensure that the goals of accessibility and expressiveness are being fully met.

5.3 Real-World Applicability: A Museum Scenario

To illustrate the practical relevance of the developed tool, consider a hypothetical cultural educator working in a medieval castle, such as the Castle of Castro Marim in southern Portugal. The educator's objective is to create an immersive storytelling experience that communicates aspects of medieval daily life, such as trade, craftsmanship and local culture, to diverse audiences.

Using the no-code authoring environment, the educator builds a narrative journey that guides visitors through scenes involving characters like a templar knight, a blacksmith or a fish vendor. These interactions are structured through the node-based interface and choreographed with platform-agnostic semantics, then exported to both AR and VR formats without rewriting the core story logic.

This cross-platform deployment enables flexible modes of engagement:

- **On-site visitors** can use their smartphones to explore the physical environment through augmented reality, scanning markers or walking to GPS-triggered locations that reveal digital content anchored in real-world spaces. This enhances traditional visits by layering interactive narratives onto physical landmarks.

- **Off-site or indoor audiences** such as classroom students or museum visitors during extreme weather conditions can access the same content through virtual reality. This mode allows them to explore a reconstructed version of the castle from within a VR headset, offering a comfortable and equally immersive alternative. For example, during the peak tourist season, when high temperatures can make outdoor exploration difficult, the VR version provides a practical way to deliver the same storytelling experience indoors.

The ability to reuse the same authored content across AR and VR maximizes reach and ensures consistent messaging. Educators can deliver coherent storytelling across multiple contexts, such as museum visits, school programs, or virtual exhibits, without duplicating authoring effort.

Chapter 6

Conclusions

This dissertation set out to explore how immersive narratives could be authored by non-technical users and deployed across both AR and VR platforms, using the concept of virtual choreographies as a unifying semantic abstraction. Starting from a previously developed AR-focused authoring tool, the project evolved through iterative design cycles informed by Design Science Research Methodology.

The first iteration expanded platform scope by introducing VR support, leading to the initial concept of dual-mapping triggers. While functional, this approach introduced redundancy and blurred the separation between narrative logic and platform implementation. The following iteration addressed this by introducing a three-file architecture composed of a platform-independent manifest, a platform-specific manifest, and a choreography file representing the narrative, enabling cleaner abstraction and scalability.

In parallel, the tool's interface was redesigned to accommodate the mapping of characters, locations, and triggers across AR and VR contexts. The player component was also enhanced to interpret these abstractions and deliver an immersive experience tailored to each platform.

Although usability testing was initially planned, time constraints limited the evaluation phase to internal feedback cycles. However, the resulting prototype achieves the core goals outlined in the problem definition: enabling no-code, cross-platform, immersive storytelling through a flexible and extensible framework.

6.1 Contributions

This dissertation builds upon the no-code authoring tool developed by Freitas [20], extending its functionality to support platform-agnostic immersive storytelling through the concept of virtual choreographies. The primary contributions of this work are as follows:

- **Three-File Manifest Architecture for Platform Abstraction:** One of the most significant contributions is the introduction of a three-file model that separates story semantics from platform implementation. By distinguishing between abstract entities, platform-specific mappings, and narrative flow, this architecture reduces redundancy, promotes modularity,

and enhances cross-platform compatibility. This design enables a single story to be deployed seamlessly in both AR and VR environments without altering its core logic.

- **Extension of the Editor for Multi-Platform Choreography Authoring:** The existing editor was extended to support this new architecture. Users can now define characters, locations, and interactions as abstract elements, and subsequently map them to either VR or AR contexts through dedicated interfaces. Tools for managing GPS anchors, QR codes, 3D object mapping, and interaction triggers were integrated into the workflow, offering a consistent and scalable method of preparing immersive experiences across platforms.
- **Integration of Platform-Agnostic Trigger System:** A unified trigger mechanism was introduced, allowing narrative nodes to reference abstract interactions that are later resolved at runtime according to the selected platform. This system enhances reusability and aligns closely with the semantic goals of virtual choreographies.
- **Visual Trigger Indicators in VR Playback:** A visual cue system was implemented in the VR player to improve story comprehension and user navigation. When a node requires user interaction, an instruction text appears and target elements are marked with rotating green octahedrons. This design decision, informed by literature on spatial storytelling and user attention, enhances immersion and aligns with best practices for non-linear, exploratory narratives.

Together, these contributions advance the field of no-code immersive content creation by introducing a scalable and semantically coherent approach to authoring platform-independent stories, rooted in the principles of virtual choreographies and supported by a concrete, extensible implementation.

6.2 Limitations

While this project succeeded in producing a functional, interoperable no-code authoring tool and associated playback system, several limitations remain. These stem both from time constraints and from deliberate decisions made to prioritize core functionality over extended features.

6.2.1 Lack of User Testing and End-User Feedback

The most significant limitation of this research lies in the absence of formal user testing. Although the system was developed with educators and cultural heritage professionals in mind, its usability and impact have not been empirically validated through structured evaluations. Usability testing, expert reviews, or pilot deployments in real-world settings like museums or classrooms would have provided critical feedback on the tool's intuitiveness, effectiveness, and storytelling potential. Without these insights, some aspects of the design, particularly around user onboarding, authoring complexity, and interaction clarity remain unverified.

6.2.2 Incomplete Functional Coverage

Several features outlined during the planning phase were not fully realized in this iteration:

- **No contextual help or dynamic tooltips:** Although the interface includes basic labeling, more comprehensive support features such as hoverable descriptions, onboarding guides, or embedded tutorials were not implemented.
- **No co-authoring or version control:** Collaborative workflows were not supported, which limits the tool's applicability for team-based or educational authoring scenarios.
- **No in-editor testing tools:** Functionality such as "start playback from selected node" or real-time simulation of story logic was not included, increasing reliance on external testing via the Player.
- **No pre-built templates:** Although planned, starter templates for common storytelling patterns were not introduced, which could have improved accessibility for beginners.

These omissions primarily affect the user experience and editor-side efficiency, which were deprioritized in favor of achieving cross-platform operability and platform abstraction.

6.2.3 Partial Platform Feature Support

Some platform-specific features were not fully maintained during this iteration. In particular:

- **3D content in AR:** While supported in earlier versions of the project, rendering 3D objects in AR was sidelined during this iteration due to structural changes in how nodes were authored and interpreted. A corresponding VR implementation of the 3D model node was never developed in the Player app, and as a result, this functionality is currently marked as "unavailable" within it. It remains available in the codebase but is not actively supported or used in the current flow.
- **Direct editor-to-player integration:** Previous implementations allowed stories to be sent directly from the editor to the player without file downloads. This feature was not preserved in the current version due to architectural and maintenance concerns, with the system now favoring a manifest-based workflow and optional combined export.

6.2.4 Extensibility Still Untested

Although the manifest-based design theoretically supports future expansion to other platforms (gesture-based VR MR, Mixed Reality...), this capability was not exercised or tested. Additionally, while VR inputs were abstracted to common interaction types (e.g., primary button, proximity), the system does not yet support variations in controller configurations or interaction schemas across VR hardware.

6.3 Future Work

Given the constraints and design decisions outlined above, the following areas have been identified as natural opportunities for future development:

- **Improved Editor Interface:** While the current interface is functional and coherent, future iterations should focus on enhancing usability, particularly for users unfamiliar with interactive narrative structures. This may include contextual tooltips, drag-and-drop refinements, or real-time node validation.
- **Expanded Action Support:** Only a core set of actions were adapted to the new system. Extending the available action types would enable more complex and varied experiences.
- **User Testing and Evaluation:** Although originally included in the work plan, formal user testing was not conducted due to time constraints. Conducting structured usability sessions with educators or non-technical creators would provide valuable feedback for refining both interface and storytelling flow.
- **Further Abstraction of Platform Controls:** While current platform manifests associate abstract interactions with predefined inputs, these mappings are still limited by assumptions about specific hardware. Future iterations could introduce a schema-based system for input abstraction, allowing developers to define how any VR controller or interaction device maps to high-level actions. This would enable the Player to interpret interaction commands dynamically based on external definitions, improving compatibility across headsets without requiring code changes.

References

[1] A-Frame. *A-Frame - Make WebVR*. 2025. URL: https://aframe.io/.

[2] R. Abadia, J. Calvert, and R. Dasika. "Effectiveness of using an immersive and interactive virtual reality learning environment to empower students in strengthening empathy and mastery learning". In: *Proceedings of the 27th International Conference on Computers in Education*. Ed. by M. Chang et al. Taiwan: Asia-Pacific Society for Computers, 2019, pp. 495–504.

[3] D. M. Adams et al. "Narrative games for learning: Testing the discovery and narrative hypotheses". In: *Journal of Educational Psychology* 104.1 (2012), pp. 235–249. DOI: 10.1037/a0025595. URL: https://doi.org/10.1037/a0025595.

[4] Adobe Aero. *Create augmented reality with AR software - Adobe Aero*. Accessed: 2025-02-24. URL: https://www.adobe.com/products/aero.html.

[5] D. Allcoat and A. von Mühlenen. "Learning in virtual reality: Effects on performance, emotion and engagement". In: *Research in Learning Technology* 26 (2018), Article 2140. DOI: 10.25304/rlt.v26.2140. URL: https://doi.org/10.25304/rlt.v26.2140.

[6] AR.js. *AR.js Documentation*. 2025. URL: https://ar-js-org.github.io/AR.js-Docs/.

[7] Articy. *Articy Draft*. Accessed: 2024-02-25. URL: https://www.articy.com/en/.

[8] A. Bhargava et al. "Evaluating multiple levels of an interaction fidelity continuum on performance and learning in near-field training simulations". In: *IEEE Transactions on Visualization and Computer Graphics* 24.4 (2018), pp. 1418–1427. DOI: 10.1109/TVCG.2018.2794639. URL: https://doi.org/10.1109/TVCG.2018.2794639.

[9] Tanat Boozayaangool. "The Design of Interplay Between Virtual Reality and Augmented Reality: A Case Study and Post Mortem of VRsus guARdian". In: *Frameless* 1.1 (2019). DOI: 10.14448/Frameless.01.008. URL: https://repository.rit.edu/frameless/vol1/iss1/21.

[10] J. Calvert and R. Abadia. "Impact of immersing university and high school students in educational linear narratives using virtual reality technology". In: *Computers & Education* 159 (2020), Article 104005. DOI: 10.1016/j.compedu.2020.104005. URL: https://doi.org/10.1016/j.compedu.2020.104005.

[11] J. Calvert and M. Hume. "Immersing learners in stories: A systematic literature review of educational narratives in virtual reality". In: *Australasian Journal of Educational Technology* 38.5 (2022), pp. 45–61. DOI: 10.14742/ajet.7032.

[12] João Cardoso. *Authoring tool for multiplatform interactive digital storytelling*. MEIC Thesis. 2023.

[13] Fernando Cassola and Maria Sousa. *XR interoperable immersive storytelling authoring tool for a medieval castle*. Accessed: 2024-10-04. 2024. URL: https://ldm.fe.up.pt/thesis/meic/2425/proposals/be5ef70e-3791-4956-858a-856e041199fb/.

[14] Fernando Cassola et al. "Design and Evaluation of a Choreography-Based Virtual Reality Authoring Tool for Experiential Learning in Industrial Training". In: *IEEE Transactions on Learning Technologies* 15.5 (2022), pp. 526–539. DOI: 10.1109/TLT.2022.3157065.

[15] Fernando Cassola et al. "Using Virtual Choreographies to Identify Office Users' Behaviors to Target Behavior Change Based on Their Potential to Impact Energy Consumption". In: *Energies* 15.12 (2022), p. 4354.

[16] CoSpaces Edu. *CoSpaces Edu for kid-friendly 3D creation and coding*. Accessed: 2025-02-24. URL: https://www.cospaces.io/.

[17] Diogo Costa. *Representation of virtual choreographies in learning dashboards of interoperable LMS analytics*. MEIC Thesis. 2023.

[18] K. Dooley. "Conceptualizing and developing narrative-based virtual reality experiences: A review of disciplinary frameworks and approaches to research". In: *Journal of Screenwriting* 14.3 (2023), pp. 229–249. DOI: 10.1386/josc_00132_1.

[19] Figma. *Figma: the Collabrative interface Design Tool*. Accessed: 2025-02-24. URL: https://www.figma.com/.

[20] Miguel Freitas. *Enabling Co-Creation for Augmented Reality: A User-Friendly Narrative Editor for Cultural Heritage Experiences*. MEIC Thesis. 2024.

[21] Vasiliki Garneli, Michail Giannakos, and Konstantinos Chorianopoulos. "Serious games as a malleable learning medium: The effects of narrative, gameplay, and making on students' performance and attitudes". In: *British Journal of Educational Technology* 48.3 (2017), pp. 842–859. DOI: 10.1111/bjet.12455. URL: https://doi.org/10.1111/bjet.12455.

[22] Daniel Green, Charlie Hargood, and Fred Charles. "Use of Tools: UX Principles for Interactive Narrative Authoring Tools". In: *Journal of Computing and Cultural Heritage* 14.3 (July 2021), pp. 1–25. DOI: 10.1145/3458769.

[23] C. Hadjipanayi et al. "Cultivating empathy through narratives in virtual reality: A review". In: *Personal and Ubiquitous Computing* 28.3–4 (2024), pp. 507–519. DOI: 10.1007/s00779-024-01812-w.

[24] Alan Hevner. "A three cycle view of design science research". In: *Scandinavian Journal of Information Systems* 19 (2007).

[25] L. Jensen and F. Konradsen. "A review of the use of virtual reality head-mounted displays in education and training". In: *Education and Information Technologies* 23.4 (2018), pp. 1515–1529. DOI: 10.1007/s10639-017-9676-0. URL: https://doi.org/10.1007/s10639-017-9676-0.

[26] F. Karakoyun and A. Kuzu. "The investigation of preservice teachers' and primary school students' views about online digital storytelling". In: *European Journal of Contemporary Education* 15.1 (2016), pp. 51–64. DOI: 10.13187/ejced.2016.15.51. URL: https://doi.org/10.13187/ejced.2016.15.51.

[27] F. Karakoyun and İ. Ü. Yapıcı. "Use of digital storytelling in biology teaching". In: *Universal Journal of Educational Research* 4.4 (2016), pp. 895–903. DOI: 10.13189/ujer.2016.040427. URL: https://doi.org/10.13189/ujer.2016.040427.

[28] E. Krokos, C. Plaisant, and A. Varshney. “Virtual memory palaces: Immersion aids recall”. In: *Virtual Reality* 23.1 (2019), pp. 1–15. DOI: 10.1007/s10055-018-0346-3. URL: https://doi.org/10.1007/s10055-018-0346-3.

[29] Demetrius Lacet, Filipe Penicheiro, and Leonel Morgado. “Magical Board Theatre: Interactive Stories That Can Be Played on Multiple Boards – Two Educational Prototypes”. In: 2021. DOI: 10.13140/RG.2.2.11068.16003.

[30] LeafletJS. *Leaflet - a JavaScript library for interactive maps*. 2025. URL: https://leafletjs.com/.

[31] C.-C. Liu et al. “An enhanced concept map approach to improving children’s storytelling ability”. In: *Computers & Education* 56.3 (2011), pp. 873–884. DOI: 10.1016/j.compedu.2010.10.029. URL: https://doi.org/10.1016/j.compedu.2010.10.029.

[32] Magnopus. *Connected Spaces Platform*. 2025. URL: https://www.magnopus.com/connected-spaces-platform.

[33] R. M. Martey et al. “Testing the power of game lessons: The effects of art style and narrative complexity on reducing cognitive bias”. In: *International Journal of Communication* 11 (2017), Article 1291. URL: https://ijoc.org/index.php/ijoc/article/view/1291.

[34] O. A. Meyer, M. K. Omdahl, and G. Makransky. “Investigating the effect of pre-training when learning through immersive virtual reality and video: A media and methods experiment”. In: *Computers & Education* 140 (2019), Article 103603. DOI: 10.1016/j.compedu.2019.103603. URL: https://doi.org/10.1016/j.compedu.2019.103603.

[35] MyWebAR. *MyWebAR by DEVAR - create augmented reality on web. Web AR for marketing, education and ecommerce*. Accessed: 2025-02-24. URL: https://mywebar.com/.

[36] Ken Peffers et al. “A design science research methodology for information systems research”. In: *Journal of Management Information Systems* 24 (2007), pp. 45–77.

[37] C. Pilegard and R. E. Mayer. “Improving academic learning from computer-based narrative games”. In: *Contemporary Educational Psychology* 44–45 (2016), pp. 12–20. DOI: 10.1016/j.cedpsych.2015.12.002. URL: https://doi.org/10.1016/j.cedpsych.2015.12.002.

[38] F. Pittarello and L. Bertani. “CASTOR: Learning to create context-sensitive and emotionally engaging narrations in-situ”. In: *Proceedings of the 11th International Conference on Interaction Design and Children - IDC ’12*. Bremen, Germany: Association for Computing Machinery, 2013. DOI: 10.1145/2307096.2307098. URL: https://doi.org/10.1145/2307096.2307098.

[39] J. E. G. Posada and M. C. C. Baranauskas. “A socio-constructionist environment to create stories using tangible interfaces”. In: *Proceedings of the 14th Brazilian Symposium on Human Factors in Computing Systems*. Salvador, Brazil: Association for Computing Machinery, 2015. DOI: 10.1145/3148456.3148457. URL: https://doi.org/10.1145/3148456.3148457.

[40] C. Y. Quah and K. H. Ng. “A Systematic Literature Review on Digital Storytelling Authoring Tool in Education: January 2010 to January 2020”. In: *International Journal of Human-Computer Interaction* 38.9 (2022), pp. 851–867. DOI: 10.1080/10447318.2021.1972608.

[41] ReactFlow. *Node-Based UIs in React - React Flow*. 2025. URL: https://reactflow.dev/.

[42] M. C. Reyes. "Measuring user experience on interactive fiction in cinematic virtual reality". In: *International Conference on Interactive Digital Storytelling*. Ed. by R. Rouse, H. Koenitz, and M. Haahr. Dublin: Springer, Dec. 2018, pp. 295–307.

[43] Tiago Ribeiro. *StoryWeaver_AR-VR: Interactive Storytelling Authoring Tool for AR/VR*. 2025. URL: https://github.com/TiagoMRib/StoryWeaver_AR-VR.

[44] S. Rizvic et al. "Actors in VR storytelling". In: *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. Vienna, Austria: IEEE, 2019, pp. 1–8. DOI: 10.1109/VS-Games.2019.8864520.

[45] J. Sadauskas, D. Byrne, and R. K. Atkinson. "Mining memories: Designing a platform to support social media based writing". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. Seoul, Republic of Korea: Association for Computing Machinery, 2015.

[46] ShapesXR. *ShapesXR — Bring ideas to life in 3D and XR*. Accessed: 2025-02-24. URL: https://www.shapesxr.com/.

[47] V. Sheafer. "Using digital storytelling to teach psychology: A preliminary investigation". In: *Psychology Learning & Teaching* 16.1 (2016), pp. 133–143. DOI: 10.1177/1475725716685537. URL: https://doi.org/10.1177/1475725716685537.

[48] Emanuel Silva, Nuno Silva, and Leonel Morgado. "Model-Driven Generation of Multi-user and Multi-domain Choreographies for Staging in Multiple Virtual World Platforms". In: *Model and Data Engineering*. Ed. by Yamine Ait Ameur, Ladjel Bellatreche, and George A. Papadopoulos. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 77–91.

[49] Andrei Torres et al. "Moirai: A No-Code Virtual Serious Game Authoring Platform". In: *Virtual Worlds* 1.2 (2022), pp. 147–171. DOI: 10.3390/virtualworlds1020009.

[50] Isabelle Verhulst et al. "Do VR and AR versions of an immersive cultural experience engender different user experiences?" In: *Computers in Human Behavior* 125 (2021). DOI: 10.1016/j.chb.2021.106951.

[51] Wonda. *Wonda - Modernize your trainings with conversational AI simulations and assessments*. Accessed: 2025-02-24. URL: https://www.wondavr.com/.

Appendix A

Example: Story Editor Output (Default Manifest File)

Listing A.1: Default Manifest JSON Example

```
{
  "title": "Castro Marim Medieval",
  "projectId": "bebc267d-97b7-42fe-9cdd-cd538e5b55a7",
  "characters": [
    {
      "id": 0,
      "name": "Narrator",
      "description": "The story's narrator",
      "image": {
        "inputType": "url",
        "filename": "../assets/character_dialogue_node.png"
      }
    },
    {
      "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
      "name": "Blacksmith",
      "description": "Works the metal",
      "image": {
        "inputType": "file",
        "filename": "64db06bf-9574-4b39-93a5-06ffdc7aecd9blacksmith.png"
      }
    },
    {
      "id": "67694b67-b1e9-416e-a663-74cadc5148bf",
      "name": "Fisherman",
      "description": "Sells fish",
```

```
      "image": {
        "inputType": "file",
        "filename": "b2b8bee0-526a-4076-9698-d9cb44139417peasant.png"
      }
    }
  ],
  "locations": [
    {
      "id": "3f110c64-ca9d-409d-a3fa-ee6002694608",
      "name": "Entrance",
      "description": ""
    },
    {
      "id": "d0c3bd4f-6711-432e-8cb1-24760e986abe",
      "name": "Blacksmith Stall",
      "description": ""
    },
    {
      "id": "a31d3c12-6eaa-4527-885d-adafb02b4735",
      "name": "Fish Stall",
      "description": ""
    }
  ],
  "interactions": [
    {
      "type": "talk_to",
      "label": "Talk to"
    },
    {
      "type": "approach",
      "label": "Approach"
    }
  ]
}
```

Appendix B

Example: Story Editor Output (AR Manifest File)

Listing B.1: AR Manifest JSON Example

```
{
  "title": "Castro Marim Medieval",
  "characters": [
    {
      "id": 0,
      "name": "Narrator",
      "trigger_type": null
    },
    {
      "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
      "name": "Blacksmith",
      "trigger_type": {
        "type": "qr",
        "value": "ferreiro"
      }
    },
    {
      "id": "67694b67-b1e9-416e-a663-74cadc5148bf",
      "name": "Fisherman",
      "trigger_type": {
        "type": "qr",
        "value": "Peixeiro"
      }
    }
  ],
  "locations": [
```

```
    {
      "id": "3f110c64-ca9d-409d-a3fa-ee6002694608",
      "name": "Entrance",
      "trigger_type": {
        "type": "gps",
        "lat": 0.8639205194187838,
        "lng": 5.947889497337622
      }
    },
    {
      "id": "d0c3bd4f-6711-432e-8cb1-24760e986abe",
      "name": "Blacksmith Stall",
      "trigger_type": {
        "type": "gps",
        "lat": 20.333989431081537,
        "lng": -20.689446959734994
      }
    },
    {
      "id": "a31d3c12-6eaa-4527-885d-adafb02b4735",
      "name": "Fish Stall",
      "trigger_type": {
        "type": "gps",
        "lat": 8.293288919921686,
        "lng": 50.34345025912525
      }
    }
  ],
  "interactions": [
    {
      "type": "talk_to",
      "label": "Talk to",
      "methodAr": "qr_code"
    },
    {
      "type": "approach",
      "label": "Approach",
      "methodAr": "gps"
    }
  ]
}
```

Appendix C

Example: Story Editor Output (VR Manifest File)

Listing C.1: VR Manifest JSON Example

```
{
  "title": "Castro Marim Medieval",
  "characters": [
    {
      "id": 0,
      "name": "Narrator",
      "threeDObject": null
    },
    {
      "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
      "name": "Blacksmith",
      "threeDObject": "blacksmith"
    },
    {
      "id": "67694b67-b1e9-416e-a663-74cadc5148bf",
      "name": "Fisherman",
      "threeDObject": "Fisherman"
    }
  ],
  "locations": [
    {
      "id": "3f110c64-ca9d-409d-a3fa-ee6002694608",
      "name": "Entrance",
      "threeDObject": "StartPosition"
    },
    {
```

```
      "id": "d0c3bd4f-6711-432e-8cb1-24760e986abe",
      "name": "Blacksmith Stall",
      "threeDObject": "WeaponStand"
    },
    {
      "id": "a31d3c12-6eaa-4527-885d-adafb02b4735",
      "name": "Fish Stall",
      "threeDObject": "FishStand"
    }
  ],
  "interactions": [
    {
      "type": "talk_to",
      "label": "Talk to",
      "methodVr": "primary"
    },
    {
      "type": "approach",
      "label": "Approach",
      "methodVr": "proximity"
    }
  ]
}
```

Appendix D

Example: Story Editor Output (Choreography File)

Listing D.1: Choreography JSON Example

```
{
  "experienceName": "Castro Marim Medieval",
  "metadata": {
    "author": "Author",
    "description": "A journey through the history of the Algarve",
    "base_manifest_url": "",
    "platform_manifest_url": ""
  },
  "story": [
    {
      "id": "0",
      "action": "begin",
      "location": "Entrance",
      "goToStep": "a73bdbe7-3bda-4ed6-84f7-ed1e7d977338"
    },
    {
      "id": "5",
      "action": "end",
      "data": {
        "ending": "Blacksmith"
      }
    },
    {
      "id": "a73bdbe7-3bda-4ed6-84f7-ed1e7d977338",
      "action": "begin-dialogue",
      "goToStep": "967a87da-a721-4f36-b792-461beb8e9c00"
```

```
    },
    {
      "id": "2",
      "action": "end-dialogue",
      "goToStep": "34e54be5-ffd4-41cb-ad24-cc90c9560868"
    },
    {
      "id": "967a87da-a721-4f36-b792-461beb8e9c00",
      "action": "text",
      "actor": {
        "id": 0,
        "name": "Narrator"
      },
      "data": {
        "text": "Welcome to the Castro Marim market"
      },
      "goToStep": "5feb3013-35a4-435f-87b5-42efd2a429ef"
    },
    {
      "id": "5feb3013-35a4-435f-87b5-42efd2a429ef",
      "action": "choice",
      "actor": {
        "id": 0,
        "name": "Narrator"
      },
      "data": {
        "text": "Which stall would you like to visit?",
        "options": [
          {
            "label": "Blacksmith",
            "goToStep": "2"
          },
          {
            "label": "Fisherman",
            "goToStep": "fd8a5668-089b-46c8-ae68-49c8981d4b77"
          }
        ]
      }
    },
    {
      "id": "fd8a5668-089b-46c8-ae68-49c8981d4b77",
      "action": "end-dialogue",
      "goToStep": "34e54be5-ffd4-41cb-ad24-cc90c9560868"
    },
    {
```

```
      "id": "34e54be5-ffd4-41cb-ad24-cc90c9560868",
      "action": "text",
      "actor": {
        "id": 0,
        "name": "Narrator"
      },
      "trigger": {
        "interaction": "approach",
        "target": "Blacksmith Stall"
      },
      "data": {
        "text": "Here it is. Talk to him to learn more about his daily
            life."
      },
      "goToStep": "24617b23-26e2-45d8-a89a-8075b4804311"
    },
    {
      "id": "0ae84c44-adb3-4df5-9f69-0869a18b352b",
      "action": "text",
      "actor": {
        "id": 0,
        "name": "Narrator"
      },
      "trigger": {
        "interaction": "approach",
        "target": "Fish Stall"
      },
      "data": {
        "text": "Here it is. Talk to him to learn more about his daily
            life."
      },
      "goToStep": "2a8cc32b-c1e4-4d50-ada9-16322b87c01f"
    },
    {
      "id": "24617b23-26e2-45d8-a89a-8075b4804311",
      "action": "text",
      "actor": {
        "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
        "name": "Blacksmith"
      },
      "trigger": {
        "interaction": "talk_to",
        "target": "Blacksmith"
      },
      "data": {
```

```
        "text": "Despite the weapons behind me, most of my work involved making nails and horseshoes, and occasionally tools."
      },
      "goToStep": "a81f333e-7bfb-4fc4-b49c-d5a2b1da44f6"
    },
    {
      "id": "a81f333e-7bfb-4fc4-b49c-d5a2b1da44f6",
      "action": "text",
      "actor": {
        "id": "0291cc4e-ca64-476c-b3a8-1d474e4190fa",
        "name": "Blacksmith"
      },
      "trigger": null,
      "data": {
        "text": "Even in times of war, spearheads sold more than swords."
      },
      "goToStep": "5"
    },
    {
      "id": "2a8cc32b-c1e4-4d50-ada9-16322b87c01f",
      "action": "text",
      "actor": {
        "id": "67694b67-b1e9-416e-a663-74cadc5148bf",
        "name": "Fisherman"
      },
      "trigger": {
        "interaction": "talk_to",
        "target": "Fisherman"
      },
      "data": {
        "text": "Every morning I bring this fish from the ocean, preserved in salt. It's one of the main elements of the city's diet."
      },
      "goToStep": "7e8ffcae-9c26-411d-b14a-2032f70d2f86"
    },
    {
      "id": "7e8ffcae-9c26-411d-b14a-2032f70d2f86",
      "action": "end",
      "data": {
        "ending": "Fisherman"
      }
    }
  ]
}
```

Appendix E

Unity Helper Scripts

E.1 Scene Exporter Script

Listing E.1: Unity script that scans a VR scene for tagged locations and actors and exports their names to a JSON file

```
using UnityEngine;
using System.IO;
using System.Collections.Generic;

public class SceneExporter : MonoBehaviour
{
    [System.Serializable]
    public class SceneData { public List<string> locations; public List<string>
        actors; }

    public string fileName = "sceneData.json";

    [ContextMenu("Export Scene Data to JSON")]
    void ExportSceneData() {
        var locations = new List<string>();
        var actors = new List<string>();
        foreach (var obj in FindObjectsOfType<GameObject>()) {
            if (obj.CompareTag("Location")) locations.Add(obj.name);
            else if (obj.CompareTag("Actor")) actors.Add(obj.name);
        }
        var data = new SceneData { locations = locations, actors = actors };
        var json = JsonUtility.ToJson(data, true);
        File.WriteAllText(Path.Combine(Application.dataPath, fileName), json);
        Debug.Log("Exported: " + fileName);
    }
}
```

E.2 Scene Normal Flipper Script

Listing E.2: Unity editor script that inverts normals and triangle winding for all meshes in the scene

```
using UnityEngine;
using System.Collections.Generic;
using UnityEditor;

public class SceneNormalFlipper : MonoBehaviour
{
    [ContextMenu("Flip All Normals in Scene")]
    void FlipAllNormals()
    {
        int flippedCount = 0;

        MeshFilter[] meshFilters = FindObjectsOfType<MeshFilter>();
        Debug.Log("Found MeshFilters: " + meshFilters.Length);

        foreach (MeshFilter mf in meshFilters)
        {
            Mesh mesh = mf.sharedMesh;

            if (mesh == null)
                continue;

            // Clone mesh to avoid modifying shared asset
            Mesh clonedMesh = Instantiate(mesh);

            // Flip normals
            Vector3[] normals = clonedMesh.normals;
            for (int i = 0; i < normals.Length; i++)
                normals[i] = -normals[i];
            clonedMesh.normals = normals;

            // Flip triangle winding
            for (int submesh = 0; submesh < clonedMesh.subMeshCount; submesh++)
            {
                int[] triangles = clonedMesh.GetTriangles(submesh);
                for (int i = 0; i < triangles.Length; i += 3)
                {
                    int temp = triangles[i];
                    triangles[i] = triangles[i + 1];
                    triangles[i + 1] = temp;
                }
                clonedMesh.SetTriangles(triangles, submesh);
            }
```

```
43
44              mf.sharedMesh = clonedMesh;
45              flippedCount++;
46          }
47
48          Debug.Log($"Flipped normals on {flippedCount} mesh(es).");
49      }
50  }
```