

LUTE: A Hypertextual Mixed Reality Game Engine

Jack Brett
Charlie Hargood
<chargood,jbrett2>@bournemouth.ac.uk
Bournemouth University
Poole, Dorset, United Kingdom

David Millard
Yoan-Daniel Malinov
Bob Rimmington
<dem,yd.malinov,e.m.rimington>@soton.ac.uk
University of Southampton
Southampton, Hampshire, United Kingdom

Abstract

Hypertext and games research have long been intertwined; both as understanding games as a form of hypertext, and also through identifying ludic interactions and patterns in hypermedia. While this research often seeks to understand theoretical and structural connections, in this paper we seek to go beyond these to build an explicitly hypertextual game engine. We target Mixed Reality (MR) and locative games for our engine as media that have been previously identified as well-suited to sculptural hypertext, and demonstrate the value of hypertextual patterns and structures as part of a game creation toolkit. In this paper we present LUTE (LoGaCulture Unity Toolkit Engine), a technology framework built on top of the Unity 3D game engine for the creation of MR games and locative experiences. LUTE builds on the established state of the art in both MR games and Creativity Support Tools (CSTs) with a modular design that uses hypertextual structures to handle the flow of content nodes in the game, and a declarative order system to specify gameplay in those nodes.

CCS Concepts

• **Human-centered computing** → **Mixed / augmented reality.**

Keywords

Mixed Reality, Game Engine, Hypertext, Locative Games

ACM Reference Format:

Jack Brett, Charlie Hargood, David Millard, Yoan-Daniel Malinov, and Bob Rimmington. 2025. LUTE: A Hypertextual Mixed Reality Game Engine. In *Proceedings of ACM Hypertext (HT '25)*. ACM, New York, NY, USA, 9 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction & Background

In recent years we have seen efforts to connect hypertext and game research by understanding hypertext as a form of game [32], and viewing games as fundamentally hypertextual experiences [33]. However, while we can understand games as hypertext *theoretically*, we are yet to explore this *practically*. In this work we move the idea of games as hypertext from merely a lens to actual architecture.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

HT '25, Chicago, US

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
<https://doi.org/XXXXXXX.XXXXXXX>

Game development can be a complex process with various interacting technologies. This can make it inaccessible to some creatives who might otherwise be contributing to the medium. This has led to the creation of a variety of “Creativity Support Tools” (CSTs) [15, 41]: technologies that help designers create games. Game engines (such as Unity 3D [46], Unreal [16], or Godot [30]) themselves could be seen as a form of CST as they blend development environments and design tools. However, there are also specialist tools for particular forms of game that leverage genre conventions to sacrifice breadth of application for more targeted creativity support. Examples include RPG Maker [13] for RPGs, Twine [27] for hypertext, or Inform [37] for text adventures. In the game narrative community, these CSTs are sometimes described as “Authoring Tools”. They use visual graph maps, domain-specific languages, or faceted interfaces to create intricate game narrative structures [7, 12, 19, 20, 28, 42].

Mixed Reality (MR) games have so far received limited support from specialist CSTs. MR games can be defined as experiences that “involve the merging of real and virtual worlds” [31] – this is in contrast to the wholly virtual worlds of virtual reality (VR). While similar to augmented reality (AR), which overlays virtual content on the real world, MR goes further, and emphasizes a two-way relationship: a virtual experience that both impacts and is impacted by physical reality. MR play experiences include: classic cultural heritage experiences such as SPIRIT [45], Viking Ghost Hunt [39], and RIOT! [8]; reconstruction works such as that by Ocura [38]; more recent museum experiences [45]; locative narratives such as the work in StoryPlaces [24, 36]; projects created by the Nottingham MR Lab [14]; and popular games such as Pokemon Go!¹.

MR games are increasingly used as serious games in the cultural heritage domain [3], where greater availability of accessible CSTs could lead to more creative diversity and positive impacts in education. A combination of technological complexity and common design patterns makes MR potentially well-suited to a bespoke CST. MR games can involve a range of complex interacting technologies: from story engines [35] to 3D assets and mixed media [11]; a variety of gameplay from puzzles [45] to navigation [39] and multiplayer [4]; location technologies [24]; and augmented visual [44] and audio [49] technology. There is also work on common structures and repeating design patterns in MR games [22, 23].

StoryPlaces presents us with an example of an existing hypertextual CST for MR experiences, with an engine for Web-based locative hypertext [24, 36]. While StoryPlaces is a generic, accessible platform, it is restricted to simpler forms of location-triggered interactive story, rather than leveraging the greater potential of

¹Niantic, 2016

extended game mechanics and play. Its CST, while novel and functional, also presents some UX design issues [26].

In this paper, we seek to address these limitations and present LUTE (LoGaCulture Unity Toolkit/Engine)²

—a hypertextual engine and CST for MR games built as a total conversion plugin for the existing Unity game engine. LUTE provides an engine for MR games in the form of a manager for the flow of content and modules, with the aim of facilitating common technologies and mechanics. It does this through a visual authoring toolkit designed for a “technical designer” who may have some familiarity with game technology, but not necessarily a high level of programming skill. LUTE is novel in that it presents a game as a hypertextual structure of *nodes* which in turn each have declarative *orders* that define the content for that node.

Existing work has already explored the potential of hypertextual structures in games [32, 33]. In particular, *sculptural hypertext* has been previously discussed as a useful approach to structuring MR experiences [23]. Traditional hypertextual structures (also known as *calligraphic hypertexts*) model content as nodes connected by links that a designer must specify in advance. *Sculptural hypertext* instead works by associating content with conditions (which check state) and behaviors (which change state) [5]. One model draws links (calligraphic), the other prunes them with state conditions (sculptural). It has been argued that sculptural hypertext is a good choice for MR due to its responsive rather than prescriptive approach—meaning that it deals well with interactions that come from real world context (such as physical location changes) [24].

Sculptural hypertext is similar to work from the interactive fiction community on ‘Storylets’ [29], in which a game is built of a content and choice state machine. In some approaches, such as StoryNexus³, the sculptural framework is mixed with calligraphic structures to give flexibility to respond to the actions of exploring players. For example, as a person navigating a virtual space to trigger content (sculptural) while specifying specific interactions for the player to make with that content, such as a character conversation tree (calligraphic) [33]. With LUTE we have also adopted this sculptural structure but extend the calligraphic elements into declarative game content. This enables the system to respond to the varied mixed reality interactions of the user, so a designer can specify changes to game state, render media, or present game mechanics as part of a state-based hypertext that responds to the player.

This paper presents LUTE as a novel use of hypertext in creating a MR game engine and CST. In doing so we are extending work that has understood games through the lens of hypertext by now building games through the technology of hypertext. We see our work as part of an established tradition of systems papers at ACM Hypertext [1, 7, 10, 24]. In the following sections, we will present LUTE and also demonstrate the potential of the tool through examples of games created with the technology, before concluding with a discussion of its novelty and the potential of this approach.

2 The Hypertextual Design of LUTE

Content within a game can flow in a mixture of sequential and open structures. Sections of play such as conversation trees in

adventure games, or levels in a puzzle game, or encounters in a role-playing game (RPG), fall one after another in pre-planned sequences either linearly or in branching trees. Alternatively in an open world players navigate a game environment and trigger content based on game state—dependent on where they explore rather than in a pre-defined sequence. Both of these patterns map neatly to calligraphic and sculptural hypertext [5] respectively. This was formalised in the Canyons-Deltas-Plains (CDP) model, which identified the presence of three macro structures in locative games (linear ‘Canyons’, branching ‘Deltas’, and open ‘Plains’) [35].

StoryPlaces implemented all three CDP patterns through sculptural hypertext [24]. While this universally sculptural approach is very robust it can also make authoring linear content clumsy as whole sequences of state checks and changes must be written. LUTE builds on this idea in that games are made out of Nodes that can be structured sculpturally, but it *also* allows canyons and deltas to be defined using calligraphic structure *within these sculptural spaces*. It therefore gets the best of both approaches: a robust, responsive design, within which it is still easy to create linear sequences.

LUTE’s core design philosophy stems from our research into the state of the art in authoring technologies for mixed reality [9] and co-design with practitioners. It can be summarized as:

- (1) **Hypertextual structure:** our games handle patterns of content using both calligraphic and sculptural hypertext.
- (2) **Accessible visual CST:** Our authoring tool wants to support creativity and bring the medium to the maximum number of designers; it is an easy-to-use visual editor.
- (3) **Flexible modularity at the mechanical level:** game content is built from modules that represent the assets and mechanics of the game. Our tools are flexible, allowing custom objects, and direct code access for expert users.

To achieve this we have a system architecture that is a conversion plugin for the Unity game engine. LUTE sits on top of Unity with its own custom interface and objects, but the games created in LUTE are Unity projects and expert users can directly edit them in Unity if necessary. The architecture can be summarized as follows:

- The hypertextual **Flow Engine** manages the structure of the game through a network of *Nodes*, and *Variables*.
 - **Nodes** represent a scene, level, or other discrete section of game content. They contain *Orders*.
 - * **Orders** instruct the game engine to render game assets, present interactions, handle game mechanics, and deliver the content of a Node.
 - **Variables** track game state and enable the *Flow Engine* to control the logical flow of *Nodes* in links and conditions, gameplay in *Orders*, and player location for MR gameplay.

2.1 The Flow Engine and Nodes

The calligraphic and sculptural structures of Nodes that make up LUTE games are handled by its Flow Engine (as depicted later in this paper in Figure 5). In Unity engine terms this is a game object within a scene that itself contains Nodes and the links, rules, and state variables that handle their structures. Each Node represents a discrete section of gameplay as structured by the designer (for example, a scene, a level, a moment in conversation) which contains game content (as described in Section 2.2).

²LUTE is available at <https://github.com/LoGaCulture/LUTE/> as of 15/7/25

³<http://wiki.failbettergames.com/start> as of 22/4/25

The Flow Engine is the central processing class for an entire experience. While designers can utilize multiple Flow Engines typically a single Flow Engine in a single Unity 3D scene suffices. The Flow Engine maintains a static list of existing Flow Engines currently in use, automatically adding new instances to this cached list. This approach, reminiscent of the singleton pattern, allows multiple instances to operate simultaneously and be accessible statically.

The Flow Engine is derived from a custom class ‘Event Windows’ that extend the Unity Editor Window with custom interactions. This editor pattern is repeated throughout LUTE with Editor Windows for other visual editors. By abstracting each view from the Event Window we ensure re-usability (for the architecture in other parts of the interface), modularity (by centralizing input handling rather than repeating it), and flexibility (allowing for custom input handling in special cases). The Flow Engine’s user interface is designed with a focus on intuitive navigation. Users can navigate the window content using either the right-click or middle-mouse button drag operations similar to other Unity windows. The position of elements (such as Nodes) is updated relative to the ‘scrollable space’, creating the illusion of an unlimited canvas size.

The Node construct in LUTE fulfills a dual role. It enables designers to define a sequential list of behaviors (Orders) while providing a framework for specifying the execution parameters of these behaviors. The content of a Node can be summarized in three parts:

- (1) **Descriptive Attributes** such as name, description, and color, for designer organization within the editor.
- (2) **Structural Behavior** wherein we find LUTE’s links—each as a bespoke Unity Event Handler. Each broadcast event serves as a link anchor, and the handler it triggers is associated with a Node. Through chains of Unity events and handlers we can create linked calligraphic structures or open sculptural sections triggered on given states. Nodes also contain attributes, controlling whether they can be revisited, and whether they are persistent and may be stored when a game is saved.
- (3) **Content** as a series of “Orders,” which constitute the gameplay and visuals of the Node as described in Section 2.2.

LUTE provides a variable system to manage state (stored in the flow engine) and support the sculptural patterns that determine which Nodes are fired. Variables can be set and updated by Orders based on player behavior, allow for a variety of data types, and support conditional statements with comparative and logical operators similar to other hypertext engines such as StoryPlaces [24]. Locative play demands managing player location as a variable, with specific Nodes restricted to locations. LUTE’s location variables are GPS radii that Node conditions can check for, and the editor handles with a custom map window as shown in Figure 1.

Within the flow editor relations between nodes are shown as lines between those nodes, these lines are always derived from the structural behaviour orders within the nodes. There are two types:

- **Calligraphic Links** - shown as solid lines, these tightly connect two Nodes together such that when the first Node completes the second is automatically triggered (using one or more ‘Next Node’ orders).
- **Sculptural Links** - shown as dashed lines, these loosely connect two Nodes such that when the first Node completes the second becomes available (using ‘Unlock Node’ orders).

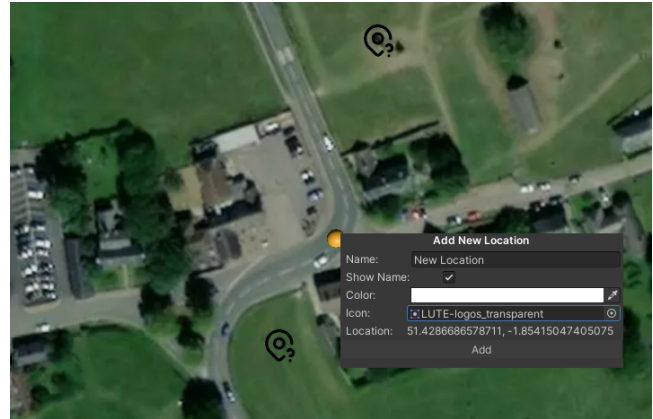


Figure 1: Creating in new location in LUTE use the map.

Nodes can contain multiple Next or Unlock Node orders, and which ones ultimately fire will depend on player actions within those nodes (for example, picking a particular dialogue choice). In this way both canyons and deltas are supported.

LUTE’s editor CST includes a number of features for the Flow Engine meant to help tackle known problems with managing structural complexity [18, 20]. It allows designers to group Nodes together within a simple graphical box as seen in figure 5. Groups of Nodes can then have conditions applied to them, or be linked in the same way as a Node, allowing for the creation of a ‘Node of Nodes’ as a form of pattern-centric authoring similar to StoryPlaces chapters [34] but handling both sculptural and calligraphic structures within each group. Groups can also be minimized, expanded, and given colors to help authors manage the visual complexity of expanding hypertext structures. LUTE’s CST also allows for annotations, both textual and graphical, to serve as a designer aid and help to organize projects and facilitate collaboration.

Finally, LUTE offers a blueprint system, where a given structure of Nodes and Orders is saved as a ‘Blueprint’ that can then be replicated elsewhere. It is supplied with some common Blueprints for the designer to use (such as a Main Menu), but designers can also create their own Blueprints for their projects, supporting what Spawforth and Millard call ‘Uncommon Patterns’ [43].

2.2 Orders and Content

While Nodes represent a hypertextual structure, within each Node the designer declares a set of Orders that represent the content for that section of the game. While classical hypertext systems work in text, and may permit content editing in text or a markup language, this is not sufficient for the complexity of rich interactive mixed media seen in locative games. Seeking accessibility, and consequently to avoid a complex domain-specific language, LUTE maintains its visual CST editor through each Node’s Order objects that declare content. These Orders instantiate game assets, media, dialogue with characters, game mechanics, augmented reality visuals, and more—and can be controlled with the same variable-based game logic as Nodes—as depicted in figure 2. When loaded by the Flow Engine, a Node executes its Orders to deliver the content of that Node to the player until the game progresses.



Figure 2: An example Order list on a Node.

Architecturally, the Order class serves as a container, specifying the Order type, name, and description. Each instance of the Order then makes reference to Unity Prefabs and Scripts that handle the actual gameplay behavior of that content, from listening for interactions to rendering graphics. LUTE comes with a variety of Orders to support common interactive fiction, adventure, puzzle, creation, and collection games seen in MR experiences. Including:

- **Characters, Dialogue, and Choice** Orders that create characters with specified sprites and names for use in other Orders. This includes Dialogue Orders that handle NPC interactions as depicted in figures 6 and 7, and Choice Orders for player responses to creative conversation trees as seen in visual novels and adventure games.
- **Inventory** Orders handle items and containers for classic adventure game item puzzles, keys and lock, and collection mechanics such as the example depicted in Figure 3.

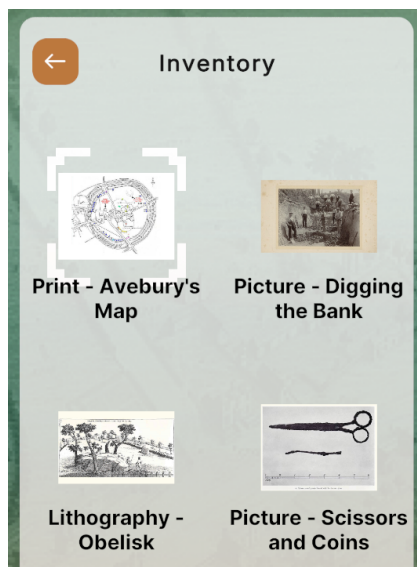


Figure 3: Example of the Inventory.

- **Canvas** Orders allow for visual creativity such as drawing, placing stickers, or other images as depicted in figure 4. These can be tied to the inventory or other game logic.



Figure 4: A simple postcard mechanic using a Canvas

- **Maps** Orders allow the game to render maps to the player, using similar functionality to the Map Editor used for creating location variables as seen in figure 1. The map can render custom pins tied to nodes and location variable and handle rules for variations on map pin sprites.
- **Interactive AR** in LUTE uses a variety of orders to use device cameras to create interactive AR Scenes. These let developers toggle AR functionality on/off, detect and track images, place objects on detected planes or locations, and manage user interactions with AR objects. The orders interact with the underlying AR Foundation Unity framework and simplify the complexities of AR development.
- **Media** functionality in LUTE is handled with its own library of Orders handling images, video, music, sound effects and more, in combination with other Orders.

These Orders have emerged from Co-Design with a community of game designers with five at first, and then later working directly with another ten to create games in LUTE as it evolved) and handle most common MR game content. In creating our hypertextual game engine we were inspired by other hypertext engines such as Twine [27] to provide an accessible visual authoring tool with common functions, but with flexibility for expert users to define and expand. As such LUTE Orders can be duplicated and customized, allowing for the user to create bespoke variations such as inventories with a different layout. They may also create entirely custom Orders using an empty container, in which they can specify any Unity functionality and content and have it work within LUTE's ecosystem. This flexibility also makes LUTE highly extensible as building on top of the framework can be as simple as adding new Orders to handle new content.

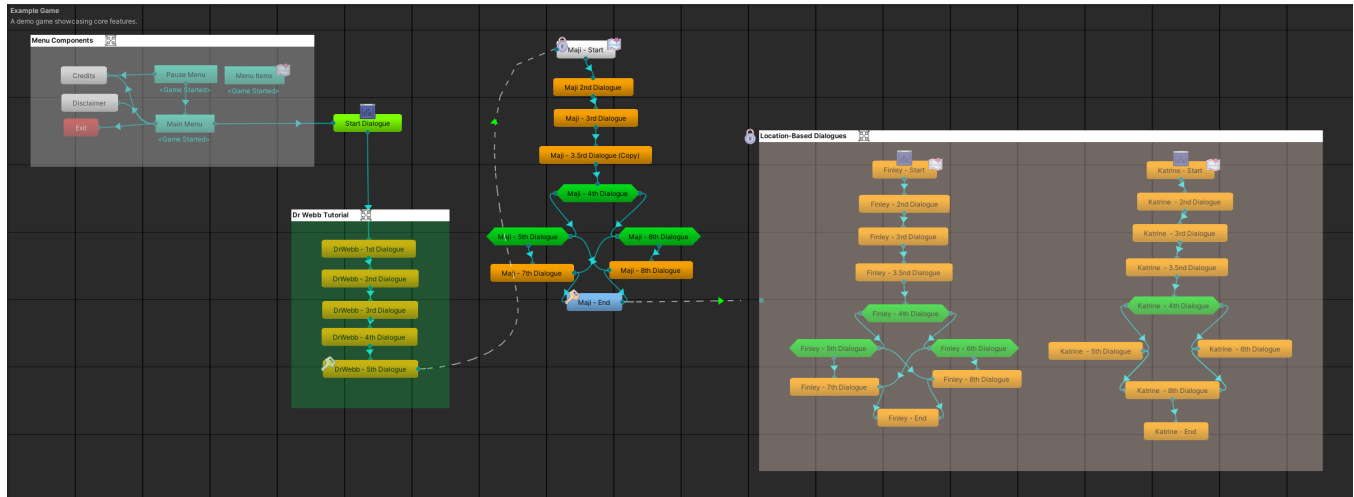


Figure 5: Example educational game constructed in LUTE (visualised on the flow engine window)

3 Demonstration

In this paper we have presented the LUTE technology framework and hypertext engine for creating MR games. A full user evaluation is outside the scope of this paper but to demonstrate the validity of the system and show its potential in this section we offer a worked example of how a game is built in LUTE, followed by three case studies of real mixed reality experiences already built in LUTE.

3.1 A Game in LUTE: A Worked Example

To illustrate and better explain the technology and hypertext engine described in Section 2 we offer a worked example. Figure 5 depicts the Flow Engine for an example LUTE game. In this simple game the core mechanics revolve around location-based exploration and interactive dialogue sequences. Players begin their journey with an introductory character (Dr Webb) who serves as a tutorial mechanism, familiarizing the player with the game’s core mechanics. As players progress through different locations, they encounter various characters who provide contextual information about the heritage site and offer rewards based on specific player interactions.

Figure 5 shows how the designer has made use of LUTE’s organizational and annotation tools, grouping sets of nodes for sections and color-coding groups and Nodes to remind them of type. Note: the icons appearing next to some Nodes indicate the presence of conditions for access based on state or location.

3.1.1 The Game Menu Flow. The game begins with a main menu with standard interface options (the white group in the top left in Figure 5). In LUTE the blueprinted ‘Main Menu’ Node incorporates ambient music and menu item orders, and exists within a cluster of Nodes governing menu-related functions. It is constructed by using ‘Menu-Choice’ Orders. When executed it identifies the first button that has not been activated and assigns the specified text and action to it. Frequently, the button press action invokes another Node, creating a classical node-link calligraphic hypertext structure in the Flow Engine. This action can also be customized, similar to any ‘On-Click’ action on standard Unity UI buttons. For example,

selecting ‘Credits’ on the main menu activates the ‘Credits’ Node, which generates a canvas object displaying credits text.

3.1.2 Tutorials. When the player starts the game from the main menu it initiates the tutorial interaction with Dr. Webb. This is represented as a linear calligraphic sequence that the designer has grouped together (the green group box on the left of Figure 5). The completion of the final Node of this sequence leads to a more complex dialogue tree with the second character, Maji—a padlock icon shows that this is the only way to access it. Unlike Dr. Webb the first of Maji’s Nodes has a location condition, which means that the player will be required to move to the specified location in order to start the interaction. An Order on the last Dr Webb Node activates the Map (with pins for available locations) to enable them to find it. During the interaction with Maji they are presented with choices (the green nodes in Figure 5). Each contains an Order that records their decision using variables.

At the end of the conversation with Maji, the sculptural part of the engine takes over, and the second pair of characters is unlocked (Finley and Katrine). These are collected into a group and therefore can be unlocked together. However, the start Nodes for each of these sequences have *both* a location condition *and* a condition that checks for the previous choice. As a result, the player must walk to a specific location in order to start the interaction and, depending on their choice while speaking with Maji, will only see one of these characters. When the encounter with Finley or Katrine begins, the Engine shifts back to calligraphic structure to handle the conversation sequence. The game utilizes three fundamental LUTE Orders within the dialogue Nodes of the NPC encounters that work together to create the interactive dialogue experience:

- **The ‘Dialogue’ Order** forms the foundation of narrative delivery, presenting text, character portraits, and contextual information to players.
- **The ‘Choice’ Order** introduces decision points within the dialogue flow. These moments allow players to shape their conversational path, offering meaningful options that can influence both immediate responses and broader narrative

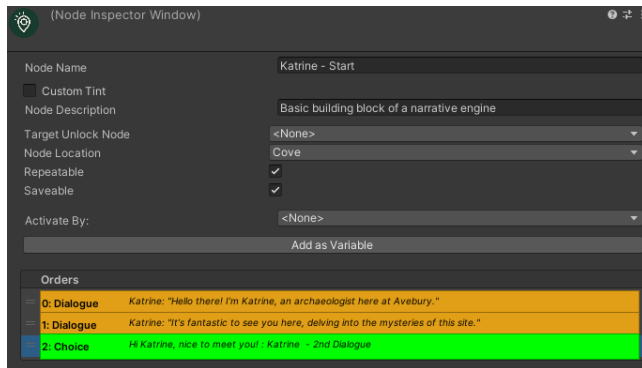


Figure 6: Editing an example Node with Dialogue and Choice Orders

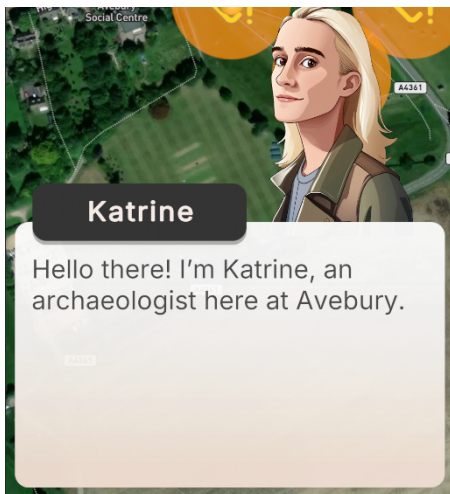


Figure 7: An example of the dialogue shown at runtime

developments. Essentially, a Choice is a button with text and a corresponding Node that is called when the Choice is selected by the player (similar to Menu Choices).

- **The 'Next Node' Order** manages the linear progression between sequential Nodes. It initiates the calligraphic navigation between the Nodes of each encounter.

Figure 6 shows one of the Nodes from the Katrine encounter. It records the location of the Node (the Cove), and shows three Orders. The first two (in orange) are 'Dialogue' Orders that display Katrine's words to the player, the last (in green) is a 'Choice'. In this case there is only one option (to return Katrine's greeting), which then leads to the next Dialogue Node. Figure 7 shows how the first of the Dialogue Orders is rendered on the screen at runtime.

This simple example shows how LUTE can manage different logical structures and enables them to be visualized to the designer. Although in this example the Nodes are very simple (demonstrating Dialogue and Choice Orders), in a larger experience they can be much more complex, such as the orders described in Section 2.2).

3.2 Case Studies

Following our worked example we can share a summary of three games built using LUTE. The games, as depicted in figure 8, have been developed with LUTE by designers exploring the cultural heritage site of Avebury Stone Circle in the UK through the medium of Mixed Reality. *Replicas* focuses on location-based exploration and quiz-based learning, while *Imagining Avebury* emphasizes creative expression through postcard design, and *Echoes of Avebury* seeks to immerse players in the world of Avebury through locative interactive storytelling and visual augmented reality (AR) 3D scenes. All 3 games include supporting artwork created by Holly Wisdom.

3.3 Game 1: Replicas

Replicas (designed by Alberto DeCaro), is the basis of our worked example in Section 3.1 (although the example shows a simplified version). It is a game in which the core mechanics revolve around location-based exploration and interactive dialogue sequences. Players learn about the heritage site and use their knowledge in assessment quiz-style questions. Upon successful completion of these assessments, players acquire virtual artifacts that are stored in the player's inventory system and can be 'used' at any time to read more information about locations they have visited previously. *Replicas* was developed entirely in LUTE without the use of coding, using Orders like Dialogues, Choices, audio players, and custom menus/buttons. The game's design incorporates calligraphic structures (dialogue trees) and sculptural structures (unlocking Nodes based on tutorials and showing specific dialogue trees dependent on player choices).

Replicas demonstrates well the potential of the mixed calligraphic-sculptural hypertext approach of LUTE. Its characters and encounters are spread across the site to be explored at the whim of the player, with some constraints based on game state—a classic locative approach that fits the state machine approach of sculptural hypertext. But the encounters themselves are more structured conversation trees with branching choices that better fit the explicit links of calligraphic hypertext. LUTE's blend of these approaches enables designers using it to develop a mixture of controlled scenes and open exploration.

3.4 Game 2: Imagining Avebury

Imagining Avebury (designed by Eloise Best), is a location-based creative game where players explore historic Avebury and build a creative memento of their day. As players navigate, they collect stickers that can be used to design their own personalized postcards.

The game uses Canvas Orders to allow players to customize their postcards with hand-drawn objects, custom text, and stickers they collect as they explore. Players can then share their creations with friends and family, or post them online. Achievements in the game are predicated by the number of postcards created, as well as their characteristics (such as total number of stickers used, and the type of sticker placed on postcards).

Imagining Avebury was developed entirely in LUTE, using Orders related to dialogue, achievements, inventory, and custom menus and buttons (such as the custom postcard-creation canvas). The game's design relies heavily on location variables, which were custom defined to reflect the player's progress and the state of the

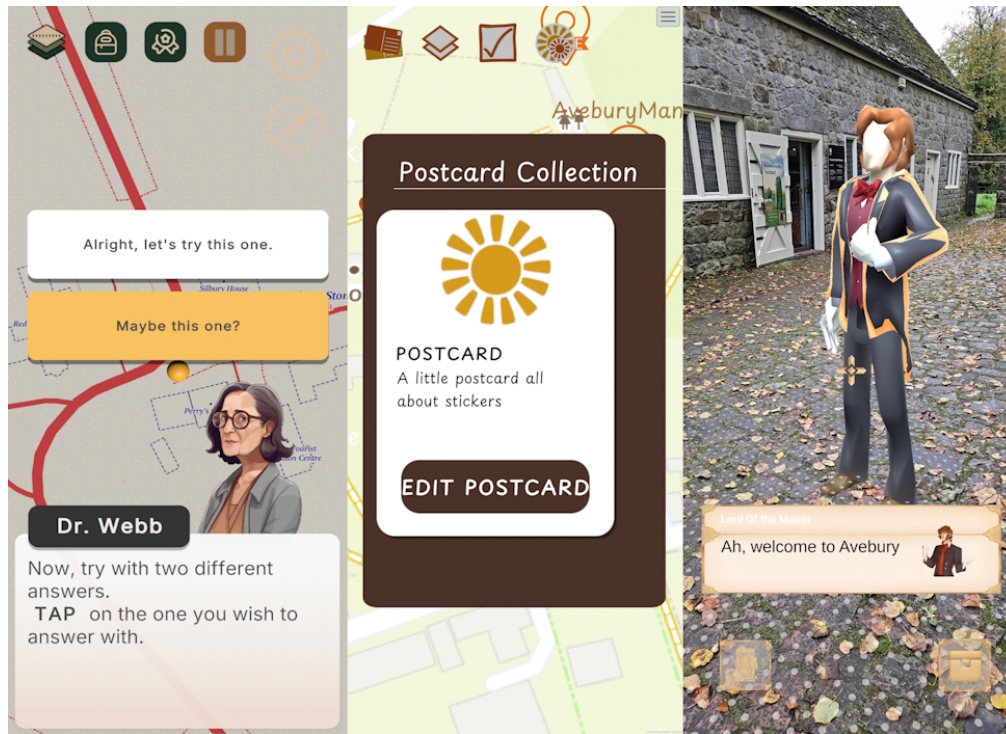


Figure 8: Screenshots from games built using LUTE: *Replicas* (left), *Imagining Avebury* (center), and *Echoes of Avebury* (right)

game. This approach allowed for a more dynamic and interactive experience, as the game’s content and progression could be tailored to the player’s individual choices and actions.

Imagining Avebury is an example of a creative MR game that is less focused on plot or objectives and more on a playful way to experience a site and create memories. LUTE’s extensible modular approach helped enable this through game mechanic Orders that interact with each other. The designer was able to use the inventory Order to build a mechanic to collect stickers around the site by visiting locations, while using the Canvas mechanic to responsively build postcards using those stickers.

3.5 Game 3: Echoes of Avebury

Echoes of Avebury (designed by Michael Smith), is an MR adventure game that immerses players in the world of Avebury, challenging them to solve a captivating murder mystery. By utilizing AR scenes, players interact with three-dimensional characters and objects in real-world locations, uncovering clues and progressing through the narrative. The game incorporates mini-games that seamlessly integrate with the overarching storyline, requiring players to solve puzzles and overcome challenges.

While the designer leveraged LUTE’s Orders for essential functionalities like dialogues, location variables, and the built-in map, they also extended the platform’s capabilities by creating custom AR scripts and logical conditions. This innovative approach showcases LUTE’s potential as a flexible tool for designers from diverse backgrounds, including those with strong programming skills, showing that LUTE is not only a front-end tool but can be used as a series of

libraries to build unique games. By combining LUTE’s foundational elements with custom-built components, the designer effectively tailored the game to the specific demands of their AR experience.

Echoes of Avebury is a good example of the power of flexible CSTs such as LUTE. The engine provided tools for the designer to build the core structures and interactions for the game, but when—as an expert Unity user—they wanted to go “off-piste” to modify custom Orders and develop their own within Unity, they were able to do so within the LUTE ecosystem without it getting in the way.

4 Discussion

This paper starts with the reflection that while ‘games through the lens of hypertext’ has been an established theoretical approach in our field, we wanted to move beyond this with our work to ‘hypertext as the *architecture* of games’. To achieve this, we took an existing 3D game engine (Unity), and built a hypertext engine on top of it (LUTE). The result is the direct application of hypertext concepts and structures, such as calligraphic and sculptural approaches, to the design and creation of games. In doing so we demonstrate the value of sculptural hypertext for open structures as proposed in earlier work [24], as well as the value of calligraphic structures in handling more sequential linear and branching content. While hypertext in more basic traditional forms is able to deliver text and images using simple text editors or a markup language, this becomes more challenging with the increased complexity of mixed reality games. While nodes and links handle structure, additional layers are required for the content of those nodes, and to maintain our visual CST we have addressed this with our modular Orders.

From a hypertext point of view LUTE is novel in the way it incorporates hypertext state into games and increases the internal complexity of nodes. LUTE could be considered as a game engine orchestrated by hypertext structure, but it could also be considered as a *strange hypertext system* with radically expanded interaction mechanics and presentation rules [6].

LUTE is also novel in how it introduces calligraphic links into a sculptural system. It does this by allowing nodes to trigger each other automatically, bypassing the sculptural checks – effectively making calligraphic sets of nodes atomic elements within the sculptural space. It also visually represents the sculptural ‘unlock’ pattern [23] as a weaker form of link. These do not trigger automatically meaning that linear or branching sequences of nodes can be placed into larger sculptural structures but remain clearly shown as sequences in their own right. This gives designers full control over tight sequences of interactions that must complete before the sculptural engine starts again (calligraphic structures) and loose sequences where the player can leave and explore other things before returning later to complete the sequence (sculptural). So far in our prototypes calligraphic sequences have been useful at the small scale (for things like dialogues with characters), whereas sculptural structures give flexibility that is useful at the large scale (such as choosing between characters).

As a toolkit LUTE is not unique in its modular approach, this is established best practice with engines such as Modulith [17] and ARGoS [40] presenting similarly modular approaches. However, both of these handle modularity at a functional component level (in terms of libraries, dependencies, and engines) rather than at a design level (in terms of LUTE’s Orders for mechanics and game content). In this sense, LUTE is closer to the visual programming systems that sit on top of some modern game engines such as Unreal’s Blueprints. However, in that case the modules are individual programming commands and functions that are much more granular than our mechanics and assets in Orders.

As such, LUTE’s approach is closer to the abstract mechanics seen in Game-o-matic [47], which also adopts a declarative approach to content. However, at this point the similarity ends as Treanor et. al’s work was focused on the procedural generation of simple news games [48] which, while very successful in its purpose, is more limited in scope than LUTE’s broader CST. In terms of modularity, LUTE sits in a gap between existing architectures in terms of both granularity and scope. Our approach is flexible rather than limited to a single form, and rather than visual programming of commands it offers visual design of game mechanics and components.

There is a taxonomic questions as to whether it is accurate to call LUTE an engine, given that it is layered on top of the existing Unity game engine as other design tools before (such as Squeezer [25] or Fungus [21]). It can be argued that LUTE’s own hypertext engine—that operates on top of Unity—makes it an engine, but this ambiguity means we present it as a Toolkit/Engine (the T and E in its name). That LUTE is both engine and CST means its approach to modularity is concerned with supporting creativity and abstracting modularity to the design level through its structures of nodes and declarative content. The approach to some extent blends the modular design of an engine with that of design toolkits such as those used by Gamers4Nature [2]. This could potentially bring some of

the accessibility and experimentation seen in interactive narrative authoring tools [19] such as Twine [27] but with the richer interactivity and play provided by a complete game engine.

LUTE is designed in such a way that its abstractions do not become an obstruction to more experienced designers that want to go beyond its tools. To the best of our knowledge, while there are simple locative experience CSTs handling basic interactive narrative or tour media content [24], LUTE is the first tool of its kind to offer full MR gameplay in an accessible visual CST.

Future work in this space has multiple opportunities. As the medium is further explored, more modular Orders can expand what is possible in LUTE, but future work might also provision for in-situ authoring [9], allowing mixed reality developers to work more in the space of their experience. Perhaps most importantly, there is also extensive user research to be conducted on the efficacy of such an approach to MR games both in terms of author experience, and the impact on resulting works. We are currently using LUTE within the LoGaCulture project, in the development of an anthology of mixed reality games (some of which we present in section 3). Iteration with our designers has led to several refinements such more sophisticated location variables and new Orders. We also have planned studies on user experience.

5 Conclusions

In this paper we have presented LUTE, a CST and platform for delivering MR games built on top of Unity. A combination of the complexity of the technology in MR games, and their commonly established design patterns, make them well suited to a CST.

Our work builds on research that uses hypertext as a lens to understand games, and goes beyond: to implement hypertext as an architecture to deliver games. Designers organize their game using a Flow Engine in terms of a mix of open *sculptural*, or more controlled *calligraphic*, hypertextual Nodes, each of which contains a declarative set of Orders specifying the content for that part of the game. This allows us to take advantage of the flexibility of sculptural structures for locative games [24], while building accessible means to declare rich game content on top—content that is focused on what the designer wants to achieve rather than how to achieve it.

Beyond our hypertextual approach, our philosophy has been one of accessibility (in terms of low barrier to use) and of flexibility. We do this by providing a visual interface that allows complicated games to be attempted by novices, but at the same time LUTE sits lightly on top of Unity such that its structures are easy to edit, extend, or add custom content to. This allows expert users to go outside of the engine’s boundaries.

Our hope is that LUTE’s Flow Engine means that the sophistication of hypertext structures can be more easily brought to mixed reality games, and that its approach to declarative content and visualization will make the creation of those games accessible to a wider group of potential designers and creators.

Acknowledgments

LoGaCulture is funded by Horizon Europe (101094036) and UKRI (10069437).

References

- [1] Claus Atzenbeck, Thomas Schedel, Manolis Tzagarakis, Daniel Roßner, and Lucas Mages. 2017. Revisiting hypertext infrastructure. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*. 35–44.
- [2] Pedro Beça, Mónica Aresta, Ana Isabel Veloso, Rita Santos, Eduardo Ferreira, Sofia Jervis, Gonçalo Gomes, Cláudia Ortet, Mariana Pereira, and Sofia Ribeiro. 2020. Developing a Toolkit to Game Design: The Gamers4Nature Project: from Concept to Artefact. In *Proceedings of the 15th International Conference on the Foundations of Digital Games (Bugibba, Malta) (FDG '20)*. Association for Computing Machinery, New York, NY, USA, Article 21, 8 pages.
- [3] Mafkereseb Kassahun Bekele, Roberto Pierdicca, Emanuele Frontoni, Eva Savina Malinverni, and James Gain. 2018. A survey of augmented, virtual, and mixed reality for cultural heritage. *Journal on Computing and Cultural Heritage (JOCCH)* 11, 2 (2018), 1–36.
- [4] Steve Benford, Andy Crabtree, Martin Flintham, Adam Drozd, Rob Anastasi, Mark Paxton, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. 2006. Can you see me now? *ACM Transactions on Computer-Human Interaction (TOCHI)* 13, 1 (2006), 100–133.
- [5] Mark Bernstein. 2001. Card shark and thespis: exotic tools for hypertext narrative. In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*. 41–50.
- [6] Mark Bernstein. 2001. Card Shark and Thespis: Exotic Tools for Hypertext Narrative. In *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia (HYPERTEXT '01)*. Association for Computing Machinery, New York, NY, USA, 41–50. doi:10.1145/504216.504233
- [7] Mark Bernstein. 2016. Storyspace 3. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media (Halifax, Nova Scotia, Canada) (HT '16)*. Association for Computing Machinery, New York, NY, USA, 201–206.
- [8] Mark Blythe, Josephine Reid, Peter Wright, and Erik Geelhoed. 2006. Interdisciplinary criticism: analysing the experience of riot! a location-sensitive digital narrative. *Behaviour & Information Technology* 25, 2 (2006), 127–139.
- [9] Jack Brett and Charlie Hargood. 2023. Authoring Tools for Mixed Reality. In *Proceedings of the 11th workshop on Narrative and Hypertext NHT' 23*.
- [10] Paul De Bra, Natalia Stash, Wouter Boereboom, Celine Chen, Joris Den Ouden, Martijn Kunstman, John Oostrum, and Egon Verbakel. 2016. ALAT: finally an easy to use adaptation authoring tool. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. 213–218.
- [11] Mara Dionisio and Valentina Nisi. 2021. Leveraging Transmedia storytelling to engage tourists in the understanding of the destination's local heritage. *Multimedia Tools and Applications* 80, 26 (2021), 34813–34841.
- [12] Henrik Engström, Jenny Bruski, and Patrik Erlandsson. 2018. Prototyping tools for game writers. *The Computer Games Journal* 7, 3 (2018), 153–172.
- [13] Enterbrain. [n. d.]. RPG Maker. <https://www.rpgmakerweb.com/> Accessed: 20/10/2024.
- [14] Martin Flintham, Steve Benford, Rob Anastasi, Terry Hemmings, Andy Crabtree, Chris Greenhalgh, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. 2003. Where on-line meets on the streets: experiences with mobile mixed reality games. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 569–576.
- [15] Jonas Frich, Lindsay MacDonald Vermeulen, Christian Remy, Michael Mose Biskjaer, and Peter Dalsgaard. 2019. Mapping the landscape of creativity support tools in HCI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–18.
- [16] Epic Games. [n. d.]. Unreal Engine. <https://www.unrealengine.com/en-US> Accessed: 20/10/2024.
- [17] Daniel Götz and Sebastian von Mammen. 2023. Modulith: A Game Engine Made for Modding. In *Proceedings of the 18th International Conference on the Foundations of Digital Games (Lisbon, Portugal) (FDG '23)*. Association for Computing Machinery, New York, NY, USA, Article 3, 8 pages.
- [18] Daniel Green. 2022. *Don't forget to save! User experience principles for video game narrative authoring tools*. Ph. D. Dissertation. Bournemouth University.
- [19] Daniel Green, Charlie Hargood, and Fred Charles. 2018. Define "Authoring Tool": A Survey of Interactive Narrative Authoring Tools. (2018).
- [20] Daniel Green, Charlie Hargood, and Fred Charles. 2021. Use of tools: UX principles for interactive narrative authoring tools. *Journal on Computing and Cultural Heritage (JOCCH)* 14, 3 (2021), 1–25.
- [21] Chris Hregen. [n. d.]. Fungus. <https://fungusgames.com/> Accessed: 20/10/2024.
- [22] Mads Haahr. 2017. Creating location-based augmented-reality games for cultural heritage. In *Serious Games: Third Joint International Conference, JCSG 2017, Valencia, Spain, November 23–24, 2017, Proceedings 3*. Springer, 313–318.
- [23] Charlie Hargood, Verity Hunt, Mark J Weal, and David E Millard. 2016. Patterns of sculptural hypertext in location based narratives. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. 61–70.
- [24] Charlie Hargood, Mark J Weal, and David E Millard. 2018. The storyplaces platform: Building a web-based locative hypertext system. In *Proceedings of the 29th on Hypertext and Social Media*. 128–135.
- [25] Mads Johansen, Martin Pichlmair, and Sebastian Risi. 2021. Squeezer - A Mixed-Initiative Tool for Designing Juice Effects. In *Proceedings of the 16th International Conference on the Foundations of Digital Games (Montreal, QC, Canada) (FDG '21)*. Association for Computing Machinery, New York, NY, USA, Article 37, 11 pages.
- [26] Sofia Kitromili, Charlie Hargood, David Millard, Huiwen Zhao, and Jim Pope. 2022. Locative Authoring: Evaluating the StoryPlaces Authoring Tool. In *Interactive Storytelling (Lecture Notes in Computer Science)*. Mirjam Vosmeer and Lissa Holloway-Attaway (Eds.). Springer International Publishing, Cham. doi:10.1007/978-3-031-22298-6_31
- [27] Chris Klimas. [n. d.]. Twine. <https://twinery.org/> Accessed: 20/10/2024.
- [28] Hartmut Koenitz. 2011. Extensible tools for practical experiments in idn: the advanced stories authoring and presentation system. In *International Conference on Interactive Digital Storytelling*. Springer, 79–84.
- [29] Max Kreminski and Noah Wardrip-Fruin. 2018. Sketching a map of the storylets design space. In *Interactive Storytelling: 11th International Conference on Interactive Digital Storytelling, ICIDS 2018, Dublin, Ireland, December 5–8, 2018, Proceedings 11*. Springer, 160–164.
- [30] Juan Limietsky and Ariel Manzur. [n. d.]. Godot. <https://godotengine.org/> Accessed: 20/10/2024.
- [31] Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems* 77, 12 (1994), 1321–1329.
- [32] David E Millard. 2020. Games/hypertext. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*. 123–126.
- [33] David E Millard and Charlie Hargood. 2021. Hypertext as a lens into interactive digital narrative. In *Interactive Storytelling: 14th International Conference on Interactive Digital Storytelling, ICIDS 2021, Tallinn, Estonia, December 7–10, 2021, Proceedings 14*. Springer, 509–524.
- [34] David E. Millard, Charlie Hargood, Yvonne Howard, and Heather Packer. 2017. The storyplaces authoring tool: Pattern centric authoring. In *Authoring for Interactive Storytelling Workshop 2017 AIS' 17*.
- [35] David E Millard, Charlie Hargood, Michael O Jewell, and Mark J Weal. 2013. Canyons, deltas and plains: towards a unified sculptural model of location-based hypertext. In *Proceedings of the 24th ACM conference on hypertext and social media*. 109–118.
- [36] David E Millard, Heather Packer, Yvonne Howard, and Charlie Hargood. 2020. The balance of attention: The challenges of creating locative cultural storytelling experiences. *Journal on Computing and Cultural Heritage (JOCCH)* 13, 4 (2020), 1–24.
- [37] Graham Nelson. 2006. Natural language, semantic analysis and interactive fiction. *IF Theory Reader* 141 (2006), 99–104.
- [38] Fumio Okura, Masayuki Kanbara, and Naokazu Yokoya. 2015. Mixed-reality world exploration using image-based rendering. *Journal on Computing and Cultural Heritage (JOCCH)* 8, 2 (2015), 1–26.
- [39] Natasa Paterson, Gavin Kearney, Katsiaryna Naliuka, Tara Carrigy, Mads Haahr, and Fionnuala Conway. 2013. Viking ghost hunt: creating engaging sound design for location-aware applications. *International Journal of Arts and Technology* 6, 1 (2013), 61–82.
- [40] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschky, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. 2012. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence* 6 (2012), 271–295.
- [41] Christian Remy, Lindsay MacDonald Vermeulen, Jonas Frich, Michael Mose Biskjaer, and Peter Dalsgaard. 2020. Evaluating creativity support tools in HCI research. In *Proceedings of the 2020 ACM designing interactive systems conference*. 457–476.
- [42] Yotam Shibolet, Noam Knoller, and Hartmut Koenitz. 2018. A framework for classifying and describing authoring tools for interactive digital narrative. In *International Conference on Interactive Digital Storytelling*. Springer, 523–533.
- [43] Callum Spawforth, Nicholas Gibbins, and David Millard. 2018. Uncommon Patterns - Authoring with Story Specific Structures. In *Authoring for Interactive Storytelling Workshop - ICIDS 2018*.
- [44] Ulrike Spierling, Jessica L Bitter, Yu Liu, and Thorolf Müller. 2023. Chances and Limitations of Immersive Augmented Reality for Game-Based Learning in Museums. In *European Conference on Games Based Learning*, Vol. 17. 643–650.
- [45] Ulrike Spierling, Peter Winzer, and Erik Massarczyk. 2017. Experiencing the presence of historical stories with location-based augmented reality. In *Interactive Storytelling: 10th International Conference on Interactive Digital Storytelling, ICIDS 2017 Funchal, Madeira, Portugal, November 14–17, 2017, Proceedings 10*. Springer, 49–62.
- [46] Unity Technologies. [n. d.]. Unity 3D. <https://unity.com/> Accessed: 20/10/2024.
- [47] Mike Treanor, Bryan Blackford, Michael Mateas, and Ian Bogost. 2012. Game-o-matic: Generating videogames that represent ideas. In *Proceedings of the The third workshop on Procedural Content Generation in Games*. 1–8.
- [48] Mike Treanor, Bobby Schweizer, Ian Bogost, and Michael Mateas. 2012. The micro-rhetorics of Game-O-Matic. In *Proceedings of the International Conference on the Foundations of Digital Games*. 18–25.
- [49] Emma Witkowski. 2018. Running with zombies: Capturing new worlds through movement and visibility practices with zombies, run! *Games and Culture* 13, 2 (2018), 153–173.