POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Development and Deployment of Civilization Evolution AR: An AR-Based Historical Learning Platform for HoloLens

Politecnico di Torino

Supervisors
Prof. Bartolomeo Montrucchio
Dr. Antonio Costantino Marceddu
Dr. Jacopo Sini

Candidate
Mengdi Yang

April 2025

Summary

Augmented reality (AR) technology is steadily becoming a part of our daily lives. In this project, we have developed an AR adaptation of a Desktop application running with Hololens glasses to create an immersive learning experience, allowing users to explore and gain deeper insights into the history of different countries. The software is currently under development, with the goal of offering an interactive platform for users to visualize the evolution of world civilizations over the centuries.

The project discussed in this thesis is part of an existing educational application initiative developed by the Department of Control and Computer Engineering (DAUIN) at Politecnico di Torino using Unity. As development continues, the software will feature more interactive capabilities to further illustrate the historical progression of global civilizations.

In the early stages of this project, two new scenarios, "1901-1950" and "1951-2000", were introduced. Each scenario includes thousands of historical events, greatly enriching the content and offering users a more comprehensive understanding of history.

The primary objective of this thesis is to deploy the existing project functionalities onto the HoloLens (VR headset), enhancing both the user experience and practical engagement.

After conducting in-depth research on the project and studying the relevant Hololens coding techniques, the migration and conversion of the project platform were gradually completed.

By replacing mouse controls with gesture-based interactions and accurately detecting finger positions, the system now displays event information when a touch event is triggered, allowing users to engage with the virtual environment more intuitively. Additionally, zoom and rotation functions for the globe have been implemented, and the menu format has been optimized to improve the overall fluidity and user experience. The event information box is also fixed on the interface, ensuring that users can easily access important details at any time during interaction, enhancing both usability and the convenience of information retrieval.

Through continuous testing, validation, and optimization, the project is being progressively refined.

The project successfully achieved its intended goals, with the deployment to the

Hololens platform now complete. This deployment not only enhanced the system's interactivity but also significantly improved the user's sense of immersion and operational experience within the virtual environment, laying a solid foundation for future optimizations and feature expansions.

By leveraging the advanced capabilities of the Hololens platform, such as hand tracking and spatial awareness, the project has taken a significant step forward in offering an immersive and engaging experience. The enhanced interactivity allows users to explore complex historical events in greater depth and interact with the content in more meaningful ways. Additionally, the improvements in the systems responsiveness and fluidity have greatly contributed to a more seamless user experience, laying the groundwork for future enhancements that can further enrich the educational potential of the software.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Professor Bartolomeo Montrucchio for giving me the opportunity to work on this project and for his valuable guidance throughout the development of this thesis.

I am also deeply grateful to Dr. Antonio Costantino Marceddu and Dr. Jacopo Sini for their consistent support, constructive feedback, and encouragement during every stage of the research. Their expertise and commitment have played a vital role in the progress and completion of this work.

Finally, I would like to thank my family and friends for their patience, understanding, and encouragement during this journey. Their support has been essential to staying focused and motivated throughout this academic experience.

Contents

List of Figures **vii**

1 Introduction **1**
1.1 Research Background 1
1.2 Research Motivation 2
1.3 Research Objectives 3
1.4 Thesis Structure Overview 3

2 Background and Related Work **5**
2.1 AR Applications in Education 5
2.2 HoloLens Technology 6
2.2.1 Key Features of HoloLens 2 7
2.2.2 Components of HoloLens 2 8
2.2.3 Applications of HoloLens 2 in Education 8
2.2.4 Relevance of HoloLens 2 to This Project 9
2.3 Previous Work 10

3 System Requirements and Design **12**
3.1 System Requirements Analysis 12
3.1.1 Functional Requirements 12
3.1.2 Non-Functional Requirements 13
3.2 System Architecture Design 13
3.2.1 Architectural Diagram 14
3.2.2 System Components 15
3.2.3 System Workflow Overview 15
3.2.4 Advantages of this Architecture 16
3.3 Technology Stack Selection 16
3.3.1 Development Platform: Unity and HoloLens 2 16
3.3.2 Framework: Mixed Reality Toolkit (MRTK) 17
3.3.3 Programming Language: C# 17
3.3.4 Data Management: Geodata and JSON-Based Storage . . . 17
3.3.5 Rendering and Graphics: DirectX and Shader Programming 18

4 Implementation **19**
4.1 System Architecture and Workflow 19
4.2 Key Modifications 20
4.2.1 Main Implementation Process 20
4.2.2 3D Globe Transformation: Scaling and Rotation 20
4.2.3 Implementation of Ray Interaction Function 22
4.2.4 Menu Format Optimization 26
4.2.5 Event Description Display 29

5 Testing and Evaluation **36**
5.1 Testing Methodology 36
5.1.1 Functional Testing 36
5.1.2 Performance Testing 37
5.1.3 User Experience Evaluation 37
5.2 Results and Analysis 38
5.2.1 Functional Testing Results 38
5.2.2 Performance Analysis 38
5.2.3 User Feedback and Improvements 39
5.2.4 Performance Improvements 41
5.3 Final Considerations 45

6 Discussion and Conclusion **47**
6.1 Technical Challenges 47
6.1.1 Hand Ray Interaction with the 3D Globe 47
6.1.2 Real-time Text Updates 48
6.1.3 Scene Transition Performance 48
6.1.4 User Learning Curve 48
6.2 Summary of Contributions 49
6.3 Limitations and Future Directions 50

Bibliography **52**

List of Figures

2.1 Geography and history education 6
2.2 Microsoft HoloLens 2 mixed reality headset 7
2.3 Previous prototype of the educational application 11

3.1 System Architectural Diagram 14
3.2 Overview of the Mixed Reality Toolkit (MRTK) features. 17

4.1 Main application interface 20
4.2 Inspector configuration diagram 21
4.3 Globe scaling and rotation gesture effect diagram 22
4.4 Ray interaction effect diagram 25
4.5 Inspector settings for the button event configuration 27
4.6 Event and city filter 29
4.7 Event and city description 35

5.1 User ratings on system usability and interaction features 40
5.2 User Guide panel displayed in HoloLens 2. 42
5.3 Async Scene Switching 45

Chapter 1

Introduction

1.1 Research Background

Augmented Reality (AR) technology [1]has undergone significant development since its inception, profoundly impacting various fields, especially education. AR was initially utilized mainly in military and industrial sectors as a technology that enhances real-world environments with computer-generated information. However, with continuous advancements in hardware, software, and interaction design, AR has gradually matured, demonstrating immense educational potential.

The development of AR has not only transformed traditional teaching methods but also enhanced interactivity and immersion in learning, enabling students to understand complex concepts more intuitively and driving innovation in educational practices.

Bulky head-mounted displays and limited graphical capabilities characterized the early stages of AR technology. With the advancement of mobile computing and smartphone technologies, augmented reality (AR) has become increasingly accessible to non-specialists. Devices such as Google Glass, Microsoft HoloLens [2], and smartphones equipped with AR capabilities have played a key role in this shift. These tools have helped bring AR into everyday contexts, including classrooms and educational environments, where its interactive potential is being actively explored.

In the field of education, augmented reality (AR) has brought new ways to improve how students learn[3]. Instead of depending only on traditional classroom methods, AR supports more active and engaging forms of learning. It encourages students to learn by doing and exploring. Through AR applications, students can:

- View complex ideas in three dimensions, which helps them better understand abstract concepts.
- Take part in interactive simulations that make theoretical knowledge easier to apply.

- Practise skills in a virtual setting, offering experiences that may not be available in a normal classroom.

This approach has been especially helpful in subjects such as science, engineering, medicine [4], and history, where students often find it difficult to understand abstract ideas using traditional methods.

ARs role in education is not limited to making lessons more interesting. It also supports different learning styles, encourages teamwork, and helps students stay focused through interactive content. In many cases, AR tools can give feedback right away, which helps learners correct mistakes and understand concepts more clearly. Research has shown that AR can help students remember information better and improve skills such as critical thinking and problem solving.

As AR technology continues to improve, its use in education is becoming more common. When combined with artificial intelligence (AI) and machine learning, AR can create personalised learning experiences that meet the needs of individual students. At the same time, as AR devices become lighter and more affordable, more schools are likely to adopt them in everyday teaching.

Over time, AR has developed from a simple tool into a powerful resource for education. As the technology becomes more advanced, it may continue to change the way teachers teach and students learn. Future research should look at how AR can be used more widely, how it can be made easier to access, and how it can fit smoothly into regular school programs.

1.2 Research Motivation

Traditional approaches to teaching history often rely on written texts, printed maps, and classroom lectures. While these methods can convey factual information, they frequently lack the interactivity and engagement necessary to fully involve students in the learning process. As a result, learners may struggle to develop a deep understanding of historical events and their broader contexts.

One key challenge lies in the static nature of traditional materials. Textbooks and maps present information in fixed formats, making it difficult for students to visualize the passage of time or understand how events connect across different periods and regions. Without dynamic or spatial representations, learners must mentally piece together fragmented knowledge, which can limit their overall comprehension.

Augmented Reality (AR) offers an innovative solution to these limitations. By presenting information in interactive 3D environments, AR helps students observe how history unfolds in real time and space. Instead of reading about abstract changes, learners can interact with historical content directly, allowing for deeper engagement and improved understanding.

This research is motivated by the potential of AR to transform how history is taught and experienced. By incorporating AR into educational settings, we aim to create a more meaningful, intuitive, and immersive way to explore the past. The use of AR can bridge the gap between learners and historical content by turning passive study into active exploration.

1.3 Research Objectives

The primary objective of this study is to improve historical learning by using Augmented Reality (AR) technology on the HoloLens 2 platform. Through this system, we aim to create an immersive and interactive learning environment that encourages student participation and enhances comprehension.

A central goal is to increase engagement by enabling users to explore history within a 3D virtual environment. The system allows learners to interact with a virtual globe, examine specific regions, and access historical information using natural gestures and gaze-based controls. This interaction model encourages curiosity and supports active learning by giving users greater control over how they navigate historical content.

Another important objective is to present historical data in a more visual and dynamic form. Unlike traditional timelines or maps, the AR system visualizes changes such as shifting borders, key battles, and the development of civilizations over time. Users can filter events by time period, type, and location, helping them form personalized and contextual views of history.

The final goal is to promote better learning outcomes by combining spatial, visual, and interactive elements. Studies suggest that multi-sensory experiences can strengthen memory and understanding. By allowing users to interact directly with content, the system encourages critical thinking and supports deeper learning.

In summary, this project aims to make history education more engaging, more visual, and more effective by using AR technology. The integration of HoloLens 2 and Mixed Reality Toolkit(MRTK)[5] provides a novel platform that enhances how learners experience and understand the past.

1.4 Thesis Structure Overview

This thesis is organized into six chapters, each focusing on a specific aspect of the research, development, and evaluation of the AR-based historical learning system. The structure of the thesis is outlined below:

- **Chapter 1: Introduction**
  This chapter introduces the project by outlining its background, motivation, and objectives. It discusses the limitations of traditional history education

and explains how AR technology may help address these issues. The chapter ends with a brief overview of the thesis layout.

- **Chapter 2: Background and Related Work**
  This chapter reviews prior work on AR in education, with particular emphasis on history-focused applications. It also presents an overview of HoloLens technology and explains its relevance to the project. The chapter concludes by identifying the key innovations of the proposed system in comparison with existing solutions.

- **Chapter 3: System Requirements and Design**
  This chapter outlines both the functional and non-functional requirements of the system. It describes the systems architecture with the aid of diagrams, and introduces major components and their workflow. The discussion also includes the rationale behind the choice of technologies such as Unity, MRTK, C#, geospatial data storage, and rendering methods.

- **Chapter 4: Implementation**
  This chapter provides a detailed account of the system development process. It starts with a summary of the systems structure and workflow, followed by key implementation elements such as coordinate transformation, globe scaling and rotation, menu interface design, and event information rendering.

- **Chapter 5: Testing and Evaluation**
  This chapter explains the methods used to evaluate the systems functionality, performance, and usability. It includes test procedures and presents the results along with analysis, aiming to assess how well the system supports interactive historical learning.

- **Chapter 6: Discussion and Conclusion**
  This chapter brings together key aspects of the systems development, highlighting technical challenges and the solutions adopted. It also discusses the main contributions of the project, current limitations, and directions for future research aimed at improving AR-based history education.

Overall, the thesis offers a well-structured investigation of the research topic, from conceptual foundation to system evaluation, providing a clear understanding of the projects significance and practical value.

Chapter 2

Background and Related Work

2.1 AR Applications in Education

Augmented Reality (AR) has emerged as a transformative technology in education, enabling the integration of digital information with the physical world. Unlike traditional learning methods, AR enriches the educational experience by overlaying interactive 3D content, animations, and contextual data onto real-world environments.

AR applications in educational settings enhance student engagement and comprehension by providing immersive, hands-on learning experiences. For instance, AR is used in:

- **Interactive Textbooks:** AR brings static content to life with 3D models and dynamic animations, making complex concepts more accessible and easier to understand.

- **Language Learning:** AR facilitates more effective language acquisition through contextual interaction by associating vocabulary with real-world objects.

- **Geography and History Education:** AR enables students to explore historical sites and geographical landmarks virtually, offering a deeper connection to the subject matter. An example can be seen in Figure 2.1.

Figure 2.1: Geography and history education

The impact of AR in education includes enhanced cognitive engagement, improved knowledge retention, and the ability to visualize abstract concepts tangibly. This project leverages AR's potential, using HoloLens to create an interactive platform that transforms traditional history education through immersive 3D visualization and real-time data interaction.

2.2 HoloLens Technology

Overview of HoloLens 2

Microsoft HoloLens 2[2] is a self-contained Mixed Reality (MR) headset that seamlessly integrates holographic content into the real world, allowing users to interact with 3D digital objects in their physical environment. As an untethered device, HoloLens 2 provides advanced interaction capabilities, enhanced computational power, and precise environmental awareness, making it particularly suitable for educational applications. Compared to its predecessor, HoloLens 2 offers improved ergonomics, a wider field of view, and enhanced hand-tracking and eye-tracking features, enabling a more immersive and intuitive user experience.

Figure 2.2 depicts the Microsoft HoloLens 2.

Figure 2.2: Microsoft HoloLens 2 mixed reality headset

2.2.1 Key Features of HoloLens 2

HoloLens 2 combines advanced hardware and software components to create a smooth and responsive mixed reality experience. The core features of the device are outlined below:

- **Intuitive Interaction Support:** The device enables natural engagement with digital content by supporting spatial mapping, hand and eye tracking, as well as voice input. These features allow users to interact with holograms in a more direct and seamless way.
- **Accurate Environmental Sensing:** It includes a dedicated Holographic Processing Unit (HPU), depth sensors, inertial measurement units (IMUs), and environmental awareness systems. Together, these components allow the system to perceive and respond to real-world surroundings in real time.
- **Immersive Visual and Audio Output:** With waveguide optics and spatial audio integration, HoloLens 2 offers high-resolution holographic visuals and 3D sound, contributing to an immersive and engaging mixed reality environment.
- **Independent Operation:** Running on the Windows Holographic operating system and supporting wireless connectivity, the device works as a standalone unit without needing external computing resources.

- **Ergonomic Design for Long-Term Use:** The headset is built with balanced weight distribution and adjustable fittings, offering improved comfort for extended periods of use in educational or workplace settings.

2.2.2 Components of HoloLens 2

The hardware system of HoloLens 2[2] is composed of several integrated components that support its mixed reality functionality. These components work together to deliver an immersive and responsive user experience:

- **Holographic Processing Unit (HPU):** A dedicated processor designed to handle spatial computing tasks such as real-time depth sensing, gesture input, and environmental interaction. It plays a central role in maintaining smooth and efficient mixed reality performance.
- **Depth and Motion Sensors:** The device includes a Time-of-Flight (ToF) camera for capturing hand movements, along with eye-tracking sensors and an Inertial Measurement Unit (IMU) that enables precise head tracking and situational awareness.
- **Waveguide Optics Display:** The display system uses holographic waveguides to present high-resolution visuals across a wide field of view. This approach helps reduce latency and supports clear, stable augmented reality imagery.
- **Spatial Mapping and Awareness System:** Multiple cameras, combined with AI-driven spatial analysis, allow the device to detect surfaces, walls, and nearby objects in real time. This capability ensures that holograms are accurately aligned with the physical environment.
- **3D Spatial Audio System:** The built-in audio system creates dynamic soundscapes that match the virtual content's position and distance, adding depth and realism to the overall experience.
- **Power and Connectivity:** HoloLens 2 is equipped with an internal battery, Wi-Fi, and Bluetooth modules, enabling mobile use without the need for external connections.

2.2.3 Applications of HoloLens 2 in Education

HoloLens 2 has found practical use in a range of educational settings, offering students immersive and interactive ways to engage with complex subjects. Its applications span several disciplines, including:

- **Medical Training:** In medical education, HoloLens 2 allows users to explore human anatomy through three-dimensional visualizations. One example is the HoloAnatomy[6] application developed by Case Western Reserve University, which enables students to examine organs, bones, and internal systems without needing physical specimens. This approach helps clarify spatial relationships between body structures and supports repeated, self-paced learning.
- **Engineering and Design:** HoloLens 2 is also used in engineering programs to support the visualization and manipulation of CAD models. At the University of Cambridge, for instance, students can view mechanical assemblies in augmented space, observe how individual components fit together, and discuss design improvements in collaborative sessions. This hands-on experience strengthens design comprehension and improves communication during team projects.
- **History and Cultural Studies:** In humanities education, the device has been applied to recreate historical and cultural environments. The Virtual Notre-Dame initiative offers a reconstructed view of the cathedral before and after the 2019 fire, allowing students to explore architectural details and understand preservation efforts. Such applications provide historical context in a more engaging and memorable format than traditional materials.

2.2.4 Relevance of HoloLens 2 to This Project

This project builds upon the interactive features of HoloLens 2 to create an AR-based system for learning history. The devices core capabilities are used in the following ways:

- **Hand Gesture and Eye Tracking:** These input methods allow users to navigate through content and interact with historical elements in a more intuitive manner.
- **Spatial Mapping and Environmental Awareness:** The system uses spatial data to anchor historical scenes within the users physical environment, helping to create a more immersive and context-aware experience.
- **Real-Time Data Integration:** Dynamic content such as animations and simulations is presented in real time, allowing users to explore historical processes as they unfold.

By incorporating these features, the project seeks to provide a more engaging alternative to traditional history instruction. Through interactive exploration, students are encouraged to develop a deeper understanding of historical events and the connections between them.

2.3 Previous Work

Prior to this research, a prototype educational application[7] was developed by the Department of Control and Computer Engineering (DAUIN) at the Politecnico di Torino. That earlier project served as the foundation for the current study. It aimed to explore the potential of virtual and augmented reality technologies for historical education, using Unity to build an interactive 3D globe interface.

The following points summarize the core features and objectives of the previous application:

- **3D Globe Interface:** Users could explore a virtual Earth model, enriched with historically significant events placed at their real-world geographic and temporal coordinates.
- **Historical Event Mapping:** Key events such as battles, scientific discoveries, cultural achievements, treaties, and technological milestones were included. Each event contained a title, short description, date, and a link to related Wikipedia articles.
- **Time Navigation:** A timeline slider allowed users to scroll backward and forward through time, in half-century increments. This feature helped users visualize how events evolved across different historical periods.
- **Geopolitical Visualization:** The system also represented the changing borders of countries over time, enabling users to observe territorial shifts, expansions, and the emergence of new nations.
- **City Information:** Each region contained structured data on cities, classified as capitals, regional capitals, or towns, to help contextualize historical developments.
- **Event Filtering:** Events could be filtered by category or theme, allowing for focused exploration of specific topics in history.
- **Color-Coded States:** A color grouping function was included to visually link related states or colonies, such as colonial powers and their territories, offering users insight into historical geopolitical dynamics.
- **Scene Expansion:** The thesis introduced two additional historical scenes, covering the periods from 1901–1950 and 1951–2000, populated with thousands of new events.

Figure 2.3 shows the prototype of the educational application, featuring a 3D globe interface that allows users to explore historical events mapped to their geographical and temporal coordinates.

Figure 2.3: Previous prototype of the educational application

These features were designed to enhance spatial and chronological understanding of historical events. The work presented in this thesis expands upon this foundation by improving interactivity, refining the user interface, and supporting more advanced educational functions through HoloLens 2 and MRTK. It aims to contribute to both classroom-based history instruction and broader research in immersive educational technologies.

Chapter 3

System Requirements and Design

3.1 System Requirements Analysis

3.1.1 Functional Requirements

To support an immersive and interactive learning experience, the system must provide several core functions:

- **3D Visualization:** The system should present interactive three-dimensional models that allow users to explore historical maps, national borders, and significant geographic features in greater detail.
- **Gesture-Based Interaction:** Instead of relying on mouse input, the system should recognize natural hand gestures, such as tapping, swiping, and pinching. Accurate finger tracking is needed to detect when users touch or select elements, enabling more intuitive interaction.
- **Event Information Display:** A fixed information panel should be included to present key historical details. This ensures that relevant data remains visible and accessible while users navigate and interact with the system.
- **Globe Manipulation:** Users should be able to smoothly rotate and zoom the 3D globe. This functionality allows them to examine different regions and time periods efficiently, supporting an exploratory approach to historical learning.
- **Optimized User Interface:** The user interface should be clear, responsive, and easy to use. This includes well-structured menus and layout elements that help maintain a smooth and consistent interaction flow.

Together, these functions help create a system that is engaging, easy to use, and effective for exploring historical information in an immersive environment.

3.1.2 Non-Functional Requirements

In addition to core functionality, the system is expected to meet several non-functional requirements related to performance, scalability, and platform compatibility, all of which contribute to a smooth and consistent user experience.

Performance

The system is designed to respond quickly to user input and maintain stable performance during interaction. The event information box remains visible at all times, allowing users to view important historical content without interruption. This layout improves usability by minimizing the need for repeated navigation. Zooming and rotating the globe should also feel smooth and responsive, with transitions occurring without noticeable lag or delay.

Scalability

To support future growth, the system architecture has been structured to handle expanding datasets and additional features. Even as the number of events or users increases, the interface should remain responsive and easy to navigate. The menu system and event display are designed to remain functional and accessible under increased demand, helping ensure the system can evolve over time without requiring major structural changes.

Compatibility

The system aims to provide a consistent experience across different devices and platforms. The interface adjusts to various screen sizes and resolutions, keeping key elements such as the event information panel visible and usable. The layout and menu structure are designed to be flexible, allowing the system to function reliably across multiple browsers and operating systems. This ensures that users can interact with the system effectively, regardless of their technical environment.

3.2 System Architecture Design

This section provides an overview of the systems architectural structure and presents a diagram to illustrate its key components.

The architecture is built to support interactive historical learning through the use of Augmented Reality (AR). It is implemented on the **HoloLens 2** platform and makes use of the **Mixed Reality Toolkit (MRTK)** to enable natural user interaction. The system is organized into three primary layers: the **Frontend (User Interaction)**, the **Data Layer**, and the **Backend Processing**. Each

layer handles a specific aspect of the systems functionality, working together to deliver a responsive and immersive experience.

3.2.1 Architectural Diagram

This section provides an overview of the systems architectural structure, highlighting the key components and their interactions. The system architecture is designed to support interactive historical learning through AR on the **HoloLens 2** platform.

The architecture is organized into several layers to ensure modularity and flexibility. These layers include the **User Interaction Layer**, **UI Layer**, **Logic Layer**, **Data Layer**, and **Rendering Layer**. Each layer handles specific tasks to optimize user experience and maintain efficient data processing.

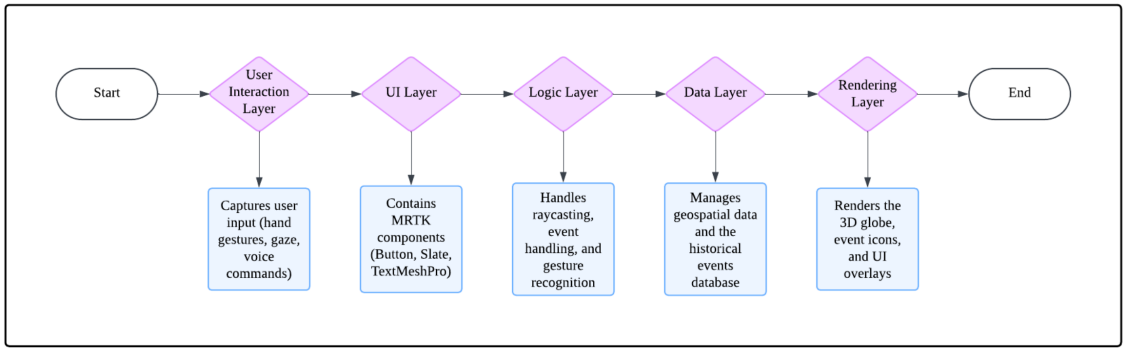

Figure 3.1: System Architectural Diagram

Figure 3.1 presents the system architecture, where the data flow follows a layered pattern.

- **User Interaction Layer**: This layer captures the user's input through HoloLens 2s sensors, including hand gestures, gaze, and optionally voice commands. It enables natural interaction with the 3D environment using raycasting, pinch gestures, and near interaction.
- **UI Layer**: Built using MRTK components such as `PressableButtonHoloLens2`, `SlateUGUI`, and `TextMeshPro`, this layer provides toggle switches for filtering data and fixed panels for displaying detailed information (e.g., event descriptions).
- **Logic Layer**: This layer handles interaction logic, including:
  - Determining intersection points between hand rays and the 3D globe.
  - Highlighting selected countries or regions.
  - Triggering the display of event or city information upon selection.

- **Data Layer**: This layer manages structured data related to countries, cities, and historical events. It connects geographic data (coordinates, boundaries) with semantic data (event descriptions, types, years).
- **Rendering Layer**: Responsible for visual output, this includes rendering the 3D Earth, event icons, region outlines, and UI overlays. Unitys rendering pipeline and TextMeshPro are utilized to ensure high performance and clear visuals on HoloLens 2.

This modular architecture ensures that the system remains scalable, maintainable, and extensible for future enhancements such as collaborative features or dynamic data loading.

3.2.2 System Components

- **Frontend Layer (User Interaction) Unity + MRTK:** The Unity Engine is the foundation for rendering the 3D world, managing AR interactions, and handling user inputs. MRTK is integrated to support **gesture-based interactions**, allowing users to rotate, zoom, and select elements on the 3D globe. The 3D Globe Visualization lets users explore historical events dynamically, while User Interface (UI) components facilitate an intuitive navigation experience.
- **Data Layer (Geodata Storage):** The system stores **geospatial data** such as country boundaries, city locations, and historical event markers, ensuring accurate historical mapping. The **Historical Event Database** organizes events by **time period, category (wars, cultural events, technological advancements), and location**. Additionally, user preferences are stored to tailor the experience.
- **Backend Processing (Interaction Logic & Event Handling):** The **Coordinate Transformation Module** converts spatial hand-tracking inputs into corresponding latitude and longitude coordinates. The **Event Management System** retrieves and updates event data dynamically based on user selections and filtering preferences. The **Gesture Recognition Module** enables **hand gesture-based zooming, panning, and event selection** via MRTK components. The Rendering Engine dynamically updates the display of event markers and UI elements.

3.2.3 System Workflow Overview

1. **Application Launch:** The user enters the **3D historical globe** environment within HoloLens 2.

2. **Scene Interaction:** Users can **rotate, zoom, and filter historical events** using hand gestures.

3. **Event Selection:** Users interact with the globe through **finger raycasting**, selecting historical events to view detailed descriptions.

4. **Data Processing & Display:** The system fetches **event details, associated images, and relevant historical timelines**, rendering them in a **fixed UI panel** for easy reading.

3.2.4 Advantages of this Architecture

- **Immersive & Intuitive:** Users can navigate history interactively instead of relying on static maps.

- **Scalable & Extensible:** New historical datasets can be integrated seamlessly without modifying core logic.

- **Optimized for AR:** Designed specifically for HoloLens 2 to utilize spatial tracking and hand interactions efficiently.

3.3 Technology Stack Selection

The selection of appropriate development tools, frameworks, and platforms is crucial for building an effective and efficient AR-based historical learning system. The chosen technology stack ensures seamless integration, optimal performance, and ease of development. The primary considerations for selection include compatibility, scalability, performance, and ease of use.

3.3.1 Development Platform: Unity and HoloLens 2

- **Unity**: As a widely adopted game and AR development engine, Unity provides comprehensive support for 3D visualization, physics-based interactions, and real-time rendering. Its built-in support for HoloLens 2 and Mixed Reality Toolkit(MRTK) makes it ideal for developing immersive AR applications.

- **HoloLens 2**: Microsoft's HoloLens 2 is one of the leading Mixed Reality (MR) headsets, offering advanced hand tracking, spatial mapping, and eye-tracking capabilities. It enables seamless AR interactions, making it highly suitable for educational applications.

3.3.2 Framework: Mixed Reality Toolkit (MRTK)

- MRTK is an open-source, cross-platform toolkit designed specifically for HoloLens and AR/VR applications.
- It provides prebuilt components for gesture recognition, UI interactions, spatial awareness, and hand tracking, significantly reducing development effort.
- The toolkit is well-maintained and optimized for performance, ensuring a smooth and natural user experience in AR applications.

Figure 3.2: Overview of the Mixed Reality Toolkit (MRTK) features.

Figure 3.2 showcases the core features of the Mixed Reality Toolkit (MRTK), including hand tracking, gesture recognition, and interactive 3D visualization, which enhance AR application development.

3.3.3 Programming Language: C#

- Unity primarily supports C#, making it the most suitable language for scripting interactions, managing game objects, and handling UI events.
- C# is object-oriented, scalable, and widely used, ensuring maintainability and ease of development.

3.3.4 Data Management: Geodata and JSON-Based Storage

- Geodata is essential for storing country borders, city locations, and historical events, ensuring accurate representation of historical information.

- JSON format stores event data, user preferences, and configuration settings, offering a lightweight and human-readable structure for easy modification and extension.

3.3.5 Rendering and Graphics: DirectX and Shader Programming

- DirectX is leveraged for optimized rendering on HoloLens 2, ensuring smooth graphics performance and real-time interactions.
- Shader programming enhances visual effects, lighting, and real-time rendering, improving the immersive experience for users.

This carefully chosen technology stack ensures the system is highly optimized, scalable, and efficient, providing an immersive and interactive historical learning experience through AR technology.

Chapter 4

Implementation

4.1 System Architecture and Workflow

To provide context and enhance understanding of the subsequent implementation details, the **core architecture** and **workflow of the system** are briefly revisited here.

The project's core architecture includes:

- **Frontend (Unity + MRTK):** Responsible for 3D interaction and UI visualization.
- **Data Layer (Geodata):** Stores country borders, city data, and historical events.
- **Interaction Logic (C#):** Handles user input, gesture recognition, and event filtering.

To facilitate understanding of the upcoming sections, the typical application usage is summarized below:

- When the user opens the application, the main scene appears (see Figure 4.1).
- The scene contains: User Guide, 3D Globe, Event and City Filter, and Event and City Description.
- Users interact with the 3D globe using hand-ray selection to explore cities and events.
- Information is displayed dynamically based on the selected event or city.

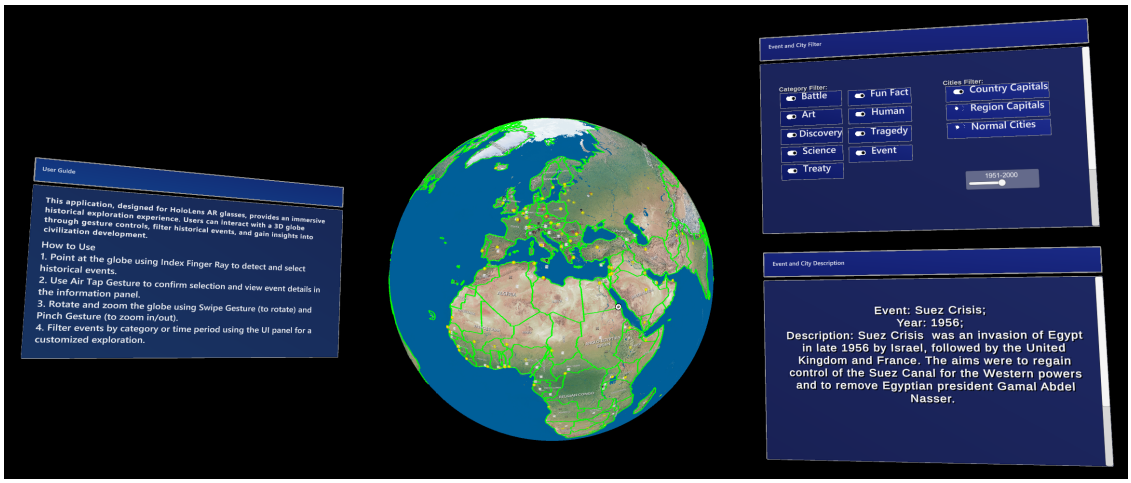

Figure 4.1: Main application interface

Figure 4.1 illustrates the main scene of the system deployed on HoloLens 2. The interface features a 3D interactive globe as the central visual element, surrounded by spatial UI panels for event filtering and detailed information display. Users can interact with the globe using natural hand gestures, select events or cities using hand rays, and view contextual information within anchored UI panels. This layout combines immersive visualization with intuitive interaction, providing a cohesive environment for exploring historical data in AR.

4.2 Key Modifications

4.2.1 Main Implementation Process

One of the core objectives of this paper is to successfully deploy the 3D Earth model on the HoloLens device and enable users to interact with it via hand gestures to select regional events. During the implementation process, the first step is to ensure that the 3D Earth model is correctly rendered and displayed in the HoloLens 2 environment, followed by the integration of ray interaction functionality.

4.2.2 3D Globe Transformation: Scaling and Rotation

In the HoloLens 2 environment, the position, scale, and rotation angle of 3D objects have a crucial impact on the user experience. To ensure that the 3D globe is positioned appropriately for viewing, its position and scale have been adjusted, slightly shifting it backward to align with the natural human visual perspective, thereby enhancing interaction convenience and immersion.

Within the Mixed Reality Toolkit (MRTK) framework, HoloLens 2 supports direct interaction with 3D objects using hand gestures. As shown in Figure 4.2,

to make the 3D globe responsive to grabbing gestures, two key components must be attached to the object: a `Collider` and an `Object Manipulator` script. The `Collider` enables the system to detect physical interactions by defining the objects boundaries, while the `Object Manipulator` controls how the object behaves during user input, including actions such as scaling, rotating, and moving the model.

Finally, the interaction configuration of the 3D globe is as follows:

- **Enable the Collider component:** Provides basic physical collision detection, allowing HoloLens 2 to recognize the presence of the object and enable interactions.
- **Add the Object Manipulator (Script):** Manages user gesture interactions with the 3D globe, including scaling, rotation, and movement.

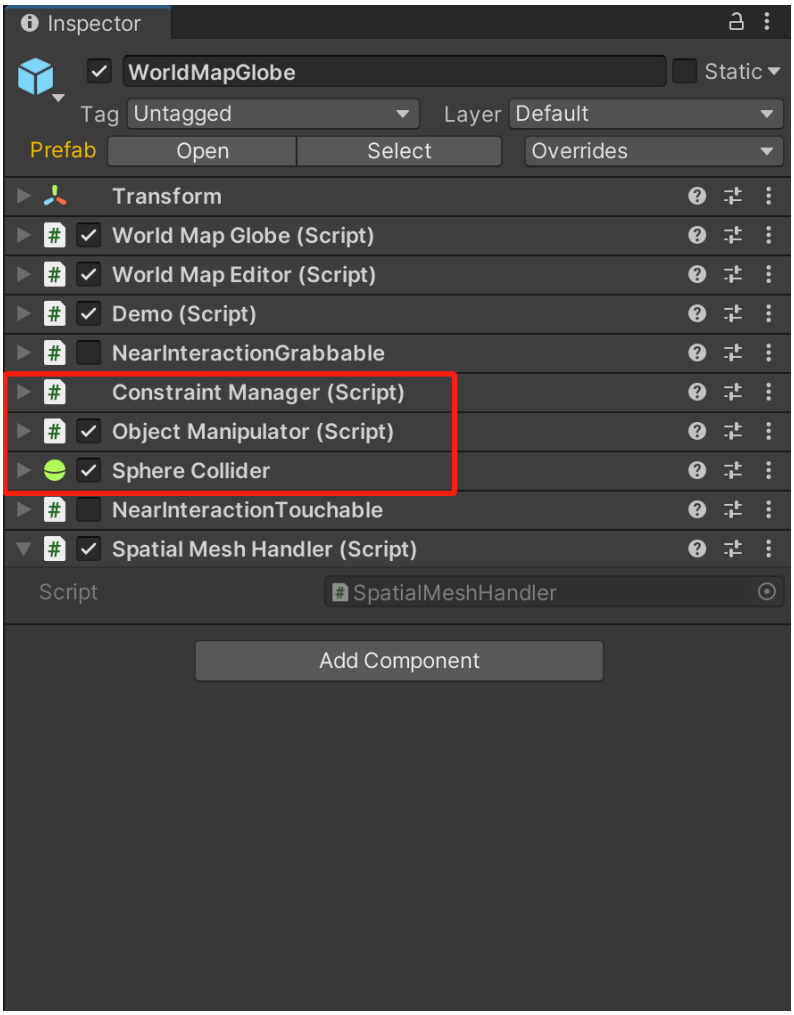

Figure 4.2: Inspector configuration diagram

The final presented effect is shown in Figure 4.3.

Figure 4.3: Globe scaling and rotation gesture effect diagram

Use the grabbing gesture (pinching with the index finger and thumb) or the ray interaction method to scale and rotate the 3D globe, enabling more flexible viewpoint adjustments.

4.2.3 Implementation of Ray Interaction Function

After successfully adjusting the visualization position and interaction methods of the 3D globe, the hand ray interaction function was further implemented, allowing users to point at and select regions and events using hand gestures. The specific process is as follows:

- **User gestures pointing at the 3D globe:** HoloLens2 allows users to generate a ray through hand gestures. The system needs to capture the direction of this ray and interact with the 3D globe accordingly.

- **Obtaining the hand ray (new feature):** A new method, TryGetHandRay(out ray), has been introduced to retrieve the ray corresponding to the user's hand gesture direction. HoloLens 2 calculates the starting point and direction of the hand ray through hand tracking, enabling users to perform pointing operations in 3D space.

  The following sections outline the newly added and modified parts of the code, implemented within the `WPMInternal.cs` script:

```
1
2  #define  VR_HOLOLENS
3
4  #if VR_GOOGLE || VR_SAMSUNG_GEAR_CONTROLLER || VR_OCULUS ||
       VR_HOLOLENS
5              _VREnabled = true; //Trigger TryGetHandRay(out ray
                   ) when _VREnabled is true.
6  #endif
7
8  #elif VR_HOLOLENS
9              if (TryGetHandRay(out ray))
10             {
11                 return true;
12             }
13
14         bool TryGetHandRay(out Ray ray)
15         {
16             ray = new Ray();
17
18             // Detect and retrieve hand input
19             foreach (var source in CoreServices.InputSystem.
                   DetectedInputSources)
20             {
21                 if (source.SourceType == InputSourceType.Hand)
22                 {
23                     foreach (var pointer in source.Pointers)
24                     {
25                         if (pointer is IMixedRealityPointer
                               mixedRealityPointer &&
                               mixedRealityPointer.Result != null)
26                         {
27                             ray = new Ray(mixedRealityPointer.
                                   Position, mixedRealityPointer.
                                   Rotation * Vector3.forward);
28                             return true;
29                         }
30                     }
31                 }
32             }
33             return false;
34         }
```

- **Calculate the intersection between the ray and the globe model:** After obtaining the ray, call the GetGlobeIntersection(out sphereCurrentHitPos) method to compute the intersection point between the ray and the 3D globe. This method, based on the spherical coordinate system of the globe, determines whether the ray intersects with the sphere and returns the three-dimensional coordinates of the intersection point.

- **Update cursor position:** Once the intersection point is calculated, the UpdateCursorLocation() method updates the cursor position, aligning it with the ray intersection point to visualize the user's interaction.

- **Detect whether the cursor is pointing at a country, determine the country index, and highlight the selected country (existing functionality):**

  The CheckMousePos() method is used to detect whether the cursor is within a specific country's region and to determine if the intersection point falls within the country's boundary.

  Then, the GetCountryUnderMouse(cursorLocation, out c, out cr) method is called to obtain the country currently being pointed at by the user and its index.

  Finally, the HighlightCountryRegion(c, cr) method highlights the selected country to enhance visual feedback, allowing the user to identify the chosen country clearly.

  A complete interaction process is achieved by integrating the detection of existing countries and highlighting functionalities.

The final presented effect is shown in Figure 4.4.

Figure 4.4: Ray interaction effect diagram

Figure 4.4 showcases 3D interaction and information visualization in the HoloLens 2 environment. Users can point at the 3D globe using a hand ray, and when the intersection point enters a country's region, the country's border turns black. At the same time, the internal area darkens to provide intuitive visual feedback. When the ray's endpoint overlaps with a city or event icon, the system automatically displays detailed information about the event.

Event information is represented using various icons, including categories such as war, art, and science, allowing users to understand different types of data quickly. This interaction method integrates spatial positioning, dynamic feedback, and layered information display, enhancing data visualization and the immersive experience. Users can freely explore significant events around the globe and utilize interactive filtering functions to customize displayed content, making data retrieval more efficient and intuitive.

4.2.4 Menu Format Optimization

In the mixed reality environment of HoloLens2, the original interaction method had certain limitations in display and operation, making it incompatible with HoloLens2 and negatively affecting the user experience. To enhance the intuitiveness and operability of interactions, this study optimized the UI design to ensure that the display of 3D objects aligns better with the visual and interaction characteristics of HoloLens2. In addition, the interaction logic was adjusted to improve the system's responsiveness and user-friendliness.

Regarding the functionality for toggling the visibility of event icons, the existing Filter Menu Toggles script was appropriately modified to better adapt to the optimized interaction logic. Users can intuitively switch the visibility of event icons via the HoloLens2 buttons, enhancing interaction convenience and clarity. Furthermore, to construct a more comprehensive interactive interface, this study employs the SlateUGUI Prefab as the Event and City Filter framework. This framework serves as a large background panel that integrates multiple ToggleSwitch buttons, enabling users to freely control the display and hiding of event and city icons through these buttons.

This design provides a more intuitive and efficient filtering interface and enhances interaction consistency and operational flexibility, making information management on HoloLens2 more convenient and effective.

Interaction Process

This study is based on the SlateUGUI Prefab as the foundation for the event and city filtering interface, combined with the PressableButtonHoloLens2ToggleSwitch component to construct a complete button press event handling process. The specific interaction steps are as follows:

1. **User's Finger Approaches the SlateUGUI Panel**
    - The SlateUGUI serves as the UI background panel and contains multiple `PressableButtonHoloLens2ToggleSwitch_32x96` controls
    - When the user's finger approaches a button`PressableButtonHoloLens2` component detects the proximity action and provides visual feedback (such as color changes and slight button movement) to indicate that an interaction is about to occur.
2. **User Presses the Button**
    - When the user presses the button, the `PressableButtonHoloLens2` component triggers the `PhysicalPressEventRouter`, which is responsible for routing the press event to the corresponding interaction logic.

3. **Button Event Trigger and Execution**

   - `PhysicalPressEventRouter` passes the event to the `Interactable` component.
   - The `Interactable` component executes the corresponding functions, including:
     - Toggling the visibility of event or city icons (by calling methods from the `Filter Menu Toggles` script).
     - Updating UI feedback (such as changing the button state or color).

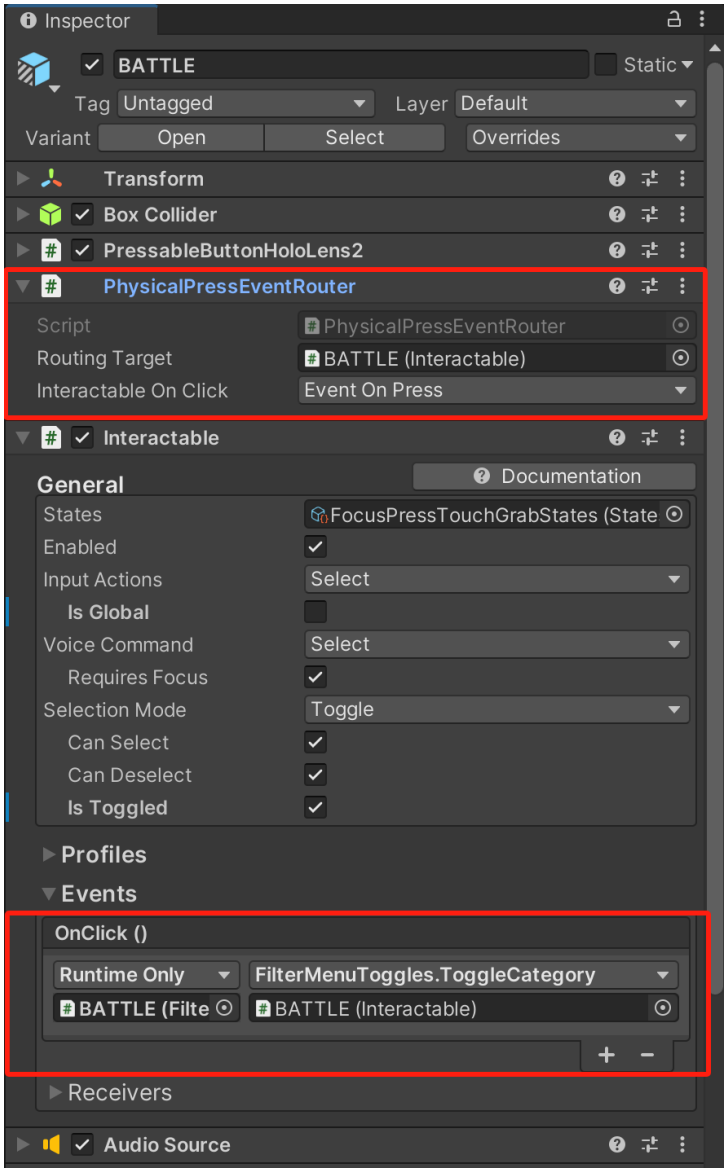

Figure 4.5: Inspector settings for the button event configuration

Figure 4.5 shows the configuration of the Battle button event in Unity's Inspector, illustrating how the button press is routed and handled through the `PhysicalPressEventRouter` and `Interactable` components.

4. **User Releases the Button**

    - `PressableButtonHoloLens2` listens for the release event and automatically restores the button to its default state, ensuring interaction stability and consistency.

Necessity of the Optimization Plan

- Introduce `SlateUGUI` as the Interactive Panel Integrates filtering buttons to make the interface more intuitive and user-friendly.

- Use `PressableButtonHoloLens2ToggleSwitch_32x96` Controls Optimizes the button interaction method to ensure the toggle functionality aligns with HoloLens 2 user expectations.

- Reuse the `Filter Menu Toggles` Script:Reduces development costs while ensuring functional stability.

- Combine `PhysicalPressEventRouter` and `Interactable` Components: Establishes a complete button event transmission logic to ensure smooth and natural interactions.

The final presented effect is shown in Figure 4.6.

Figure 4.6: Event and city filter

4.2.5 Event Description Display

In the original interaction design, event description information was displayed as a floating tooltip near the mouse cursor, and its position was dynamically adjusted as the cursor moved. While this approach provided instant feedback during user interaction, limitations such as small font size and unstable positioning could affect readability and user experience.

To optimize the information display, this study introduces the SlateUGUI Prefab as the event description display framework. Unlike the original cursor-following display, the event description is now fixed within the SlateUGUI component, achieving the following improvements:

1. **Enhancing Text Readability**
    - The font size of the event description is increased, making the text clearer and improving reading comfort.
    - The `TextMeshPro` component is used to optimize text rendering, ensuring high-quality text display at high resolutions.
2. **Enhancing Interface Stability**
    - The event description is fixed within the `SlateUGUI` interface, preventing information from shifting with mouse movements and allowing users to focus more on the content itself.

- As a standard UI component, `SlateUGUI` adapts to the 3D interactive environment of HoloLens 2, providing a more intuitive visual experience.

3. **Improving Interaction Experience**

   - The event description display area is integrated with other UI components, allowing users to browse historical event information more conveniently and perform filtering or operations.
   - By incorporating the `Filter Menu Toggles` function, users can freely select the event categories to be displayed, making the interface more customizable.

Based on the above optimization objectives, this study adopts `SlateUGUI` as the fixed display area for event descriptions and integrates the `TextMeshPro` component for text rendering optimization. This ensures a more stable, precise, and interactive information display on HoloLens 2 devices. Compared to the original cursor-following text display, this approach enhances text readability, improves interface stability, and optimizes the overall interaction experience. The following sections will detail the dynamic generation and display process of event descriptions. All related code modifications and additions have been implemented within the `demo.js` script.

1. **Dynamically Creating a `TextMeshPro` Text Object and Attaching It to `SlateUGUI`**

   To achieve a fixed display for event descriptions, the system first locates `Scroll View` as the parent object of `TextObject`, ensuring that the text is properly embedded within the UI structure. Then, `TextObject` is dynamically created and assigned as a child of `Scroll View`, allowing it to scroll along with the UI interface, ensuring continuity in information display and stability in interaction.

   The following are the modifications and additions to the code.

```
public GameObject TextObject;  //new
public TextMeshPro textMeshPro; //new
void Start()
    {
    //Cursor.SetCursor(null, hotSpot, cursorMode);

    map = WorldMapGlobe.instance;
    map.FlyToCountry("Italy");
    map.showNormalCities = false;
    map.showRegionCapitals = false;

    //new
```

```
13      GameObject parentObject = GameObject.Find("
            EventDescription/UGUIScrollViewContent/Scroll View");
14      //cam = map.mainCamera;
15      TextObject = new GameObject("MyTextObject");
16      TextObject.transform.parent = parentObject.transform;
17      textMeshPro = TextObject.AddComponent<TextMeshPro>();
```

2. **Initializing the `TextMeshPro` Component for Optimized Text Rendering**

   The `TextMeshPro` component is added to `TextObject`, with font size, color, alignment, and word wrapping properties adjusted to enhance text clarity and readability. Compared to the traditional `Text` component, `TextMeshPro` offers higher-quality text rendering, ensuring clear text display in the high-resolution environment of HoloLens 2.

   The following are the modifications and additions to the code.

```
1   if (textMeshPro != null)
2       {
3           // Set the properties of TextMeshPro
4           textMeshPro.fontSize = 280;
5           textMeshPro.color = Color.white;
6           textMeshPro.alignment = TextAlignmentOptions.Center;
7           textMeshPro.enableWordWrapping = true;
8           textMeshPro.fontStyle = FontStyles.Bold;
9           textMeshPro.text = "Hello, World!";
10
11          RectTransform rectTransform = TextObject.GetComponent<
                RectTransform>();
12          rectTransform.sizeDelta = new Vector2(750, 200);
13          rectTransform.localRotation = Quaternion.Euler(0, 0,
                0);
14          rectTransform.localPosition = new Vector3(0, 40, 0);
15          rectTransform.localScale = parentObject.transform.
                localScale * 1.0f;
16
17          TextObject.layer = LayerMask.NameToLayer("UI");
18
19          //TextObject.transform.localPosition = new Vector3(7,
                225, -105);
20          //TextObject.transform.localScale = parentObject.
                transform.localScale * 1.0f;
21          //float distanceFromCamera = 40.0f;
22          //SetDistanceFromCamera(distanceFromCamera);
23      }
24      else
25      {
26          Debug.LogError("TextMeshPro is not assigned!");
27      }
```

3. **Adjusting `RectTransform` and UI Hierarchy to Ensure Stable Text Positioning**

   To ensure proper text display within the `SlateUGUI` component, the `RectTransform` is adjusted for size, position, and rotation. The UI hierarchy of `TextObject` is also set to ensure seamless integration with other `SlateUGUI` components, preventing display issues caused by incorrect hierarchy settings. Additionally, fixing the text display area avoids unwanted text movement due to cursor motion, ensuring a more stable information presentation.

4. **Integrating `Filter Menu Toggles` and Dynamic Text Updates to Enhance Interaction Experience**

   To further optimize the interaction experience, this study integrates the event description area with `Filter Menu Toggles`, allowing users to freely filter and control the event categories displayed, thereby providing a more personalized UI experience. Meanwhile, the text content is dynamically updated to ensure users can instantly access relevant information during interaction, avoiding information overload or distractions that could affect readability.

   To achieve this functionality, the system utilizes the `TextMeshPro` component for text rendering while continuously monitoring the user's interaction status. Based on the currently selected object type, the system adjusts the `text` property of `TextMeshPro` to provide real-time feedback and enhance the interaction experience. When a user selects a city, particularly a country's capital, the system displays the city name, province, and country, such as:

   > City: Rome (Lazio, Italy)

   If the user selects an event, such as a war, the system displays the event name, year, and detailed description, for example:

   > Event: Battle of Waterloo;
   > Year: 1815;
   > Description: The battle marked the final defeat of Napoleon Bonaparte.

   This dynamic text update mechanism continuously monitors changes in `map.cityHighlighted` and `map.categoryHighlighted` within the `OnGUI()` method, instantly adjusting the display content of the `TextMeshPro` component. Compared to static text display methods, this approach enhances the clarity and intuitiveness of information presentation and improves the immersive experience during interactions in HoloLens 2. It enables users to access target information more intuitively without requiring additional operations.

   The following are the modifications and additions to the code.

```
void OnGUI()
{
    if (map.countryHighlighted != null && map.cityHighlighted
        != null || map.provinceHighlighted != null || map.
        categoryHighlighted != null)
    {
        City city = map.cityHighlighted;
        Category category = map.categoryHighlighted;
        //da rivedere
        if (city != null)
        {
            switch (city.cityClass)
            {
                case CITY_CLASS.COUNTRY_CAPITAL:
                    if (WorldMapGlobe.instance.
                        showCountryCapitals)
                    {
                        if (city.province != null && city.
                            province.Length > 0)
                        {
                            text = "City: " + map.
                                cityHighlighted.name + " (" +
                                city.province + ", " +
                                    map.countries[map.
                                        cityHighlighted.
                                        countryIndex].name + ")"
                                        ;
                        }
                        else
                        {
                            text = "City: " + map.
                                cityHighlighted.name + " (" +
                                    map.countries[map.
                                        cityHighlighted.
                                        countryIndex].name + ")"
                                        ;
                        }
                        textMeshPro.text = text;//new
                    }

                    break;
                }
            }
        else if (category != null)
            {
            switch (category.categoryClass)
            {
                case CATEGORY_CLASS.BATTLE:
                    if (WorldMapGlobe.instance.showBattles)
                    {
```

```
38                          if (category.description != null || !
                                string.Equals(category.description,
                                 "") || category.year != -1)
39                          {
40                              stringToEdit = "Event:␣" + map.
                                    categoryHighlighted.name + ";\
                                    nYear:␣" + map.
                                    categoryHighlighted.year + ";\
                                    nDescription:␣" + map.
                                    categoryHighlighted.description
                                    ; //+ ";\nLink:␣" + map.
                                    categoryHighlighted.
                                    categoryLink;
41                              //stringToEdit = GUI.TextArea(new
                                    Rect(x, y, 250, 150),
                                    stringToEdit, 300);
42                              textMeshPro.text = stringToEdit;//
                                    new
43                          }
44                          else
45                          {
46                              stringToEdit = "";
47                          }
48                      }
49                      break;
50                  }
51              }
```

Figure 4.7 displays detailed information about the Transnistria War event, including the event name, year of occurrence, and specific description, while also showcasing its visualization on the HoloLens 2 device.

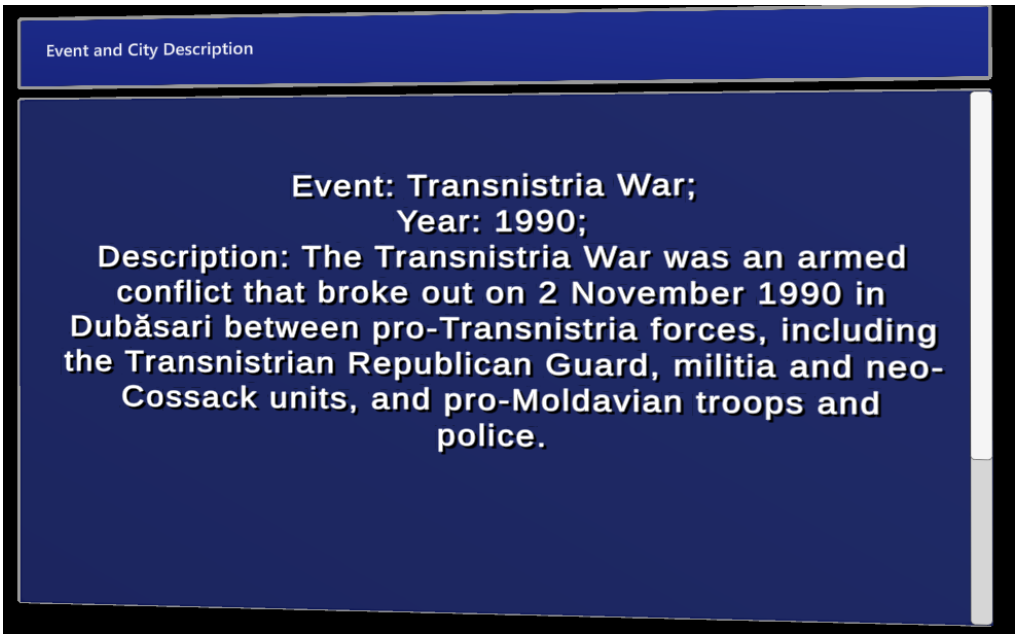

Figure 4.7: Event and city description

Chapter 5

Testing and Evaluation

5.1 Testing Methodology

To ensure the reliability, efficiency, and usability of the system, a comprehensive testing methodology was adopted. This includes functional testing to validate core features, performance testing to measure system responsiveness, and user experience evaluation to gather insights from real-world interactions.

5.1.1 Functional Testing

Functional testing was conducted to verify the correctness and stability of each major feature. The following key functionalities were tested:

- **Event Description Display:** Ensures that selecting a city or event correctly updates the displayed information in `TextMeshPro`.
- **Interaction with Hand Ray:** Verifies that users can accurately select regions and events using hand ray interactions.
- **Filter Menu Toggles:** Tests whether users can effectively filter displayed event categories and toggle visibility settings.
- **UI Stability:** Checks that event descriptions remain correctly positioned within `SlateUGUI` and do not shift unexpectedly.
- **Dynamic Text Updates:** Validates that text content updates instantly when users interact with different UI elements.

Each test case was executed multiple times to ensure consistent results, and any failures were documented for debugging and refinement.

5.1.2 Performance Testing

To assess system responsiveness and efficiency, the following performance metrics were measured:

- **Rendering Latency:** Evaluated the time taken for `TextMeshPro` content updates to reflect user interactions.
- **Frame Rate Stability:** Monitored frames per second (FPS) to ensure smooth operation under various interaction loads.
- **Memory Usage:** Analyzed the systems memory footprint to prevent excessive resource consumption.
- **Interaction Delay:** Measured the delay between user input (e.g., hand ray selection) and UI response.

Performance tests were conducted under different conditions, including varying numbers of events and cities displayed simultaneously, to identify potential bottlenecks.

5.1.3 User Experience Evaluation

This study uses a questionnaire survey method to evaluate user experience, focusing on aspects such as interface design, interaction smoothness, information readability, and overall immersion. Participants completed the survey after using the system and rated different aspects on a scale from 1 (Very Poor) to 5 (Excellent).

Survey content (excerpt):

- Is the interface design clear and intuitive?
- Are the gesture interactions smooth and natural?
- Is the information display easy to read and understand?
- How immersive is the overall experience?
- Is the application easy to learn and use?
- Is the historical era switching feature intuitive?
- How accurate is the index finger ray in selecting events?
- How smooth is the rotation and zooming of the globe?
- Is the menu and filtering function convenient to use?

Additionally, the questionnaire includes sections for issue reporting and improvement suggestions, where users can describe encountered technical problems, indicate features they like, and suggest new functionalities for future updates, such as voice control, 3D historical event visualization, and multi-user collaboration mode.

5.2 Results and Analysis

The test results provided insights into the systems effectiveness, highlighting both strengths and areas requiring improvement.

5.2.1 Functional Testing Results

The system successfully handled core functionalities, with all primary test cases passing. However, minor inconsistencies were noted in:

- Occasional delays in text updates when rapidly switching between events.
- Rare misalignment of text within the `SlateUGUI` under high interaction loads.
- During scene transitions, there is a noticeable delay in interface loading, affecting the smoothness of the user experience.

Improvement Plan: To address these issues, the system has been optimized as follows:

- Optimized the text rendering process to improve text update efficiency during event transitions.
- Adjusted the UI layout logic to minimize text misalignment under high interaction loads.
- Added a loading animation during scene transitions to mitigate loading delays, ensuring a smoother interface transition experience.

5.2.2 Performance Analysis

- The system maintained an average frame rate of **60 FPS**, ensuring smooth visual performance.
- Text update latency was measured at an average of **50ms**, which remained within acceptable limits for real-time interaction.
- Memory usage remained stable, with no significant memory leaks detected during extended usage.

- Interaction response times remained below **200ms**, allowing for fluid user experience.

Areas for improvement included further optimization of text update efficiency and resource allocation in scenarios involving high-density event markers.

5.2.3 User Feedback and Improvements

The system was tested by five participants, who interacted with its core features and shared their impressions. Overall, their feedback was positive across multiple aspects, along with several constructive suggestions for further improvement.

User Ratings:

- Interface Design: 5 (5+5+5+5+5)/5
- Gesture Interaction: 3.8 (4+5+4+3+3)/5
- Information Readability: 4.8 (5+5+5+5+4)/5
- Immersive Experience: 4 (4+3+5+4+4)/5
- Learning Curve: 3.2 (3+4+3+4+2)/5
- Era Switching Function: 5 (5+5+5+5+5)/5
- Event Selection Accuracy: 4 (4+3+4+5+4)/5
- Globe Rotation and Zooming: 3.8 (4+3+5+3+4)/5
- Menu and Filtering Convenience: 5 (5+5+5+5+5)/5

Figure 5.1 provides a visual summary of user evaluations on key system features, including interface design, gesture interaction, and overall usability.

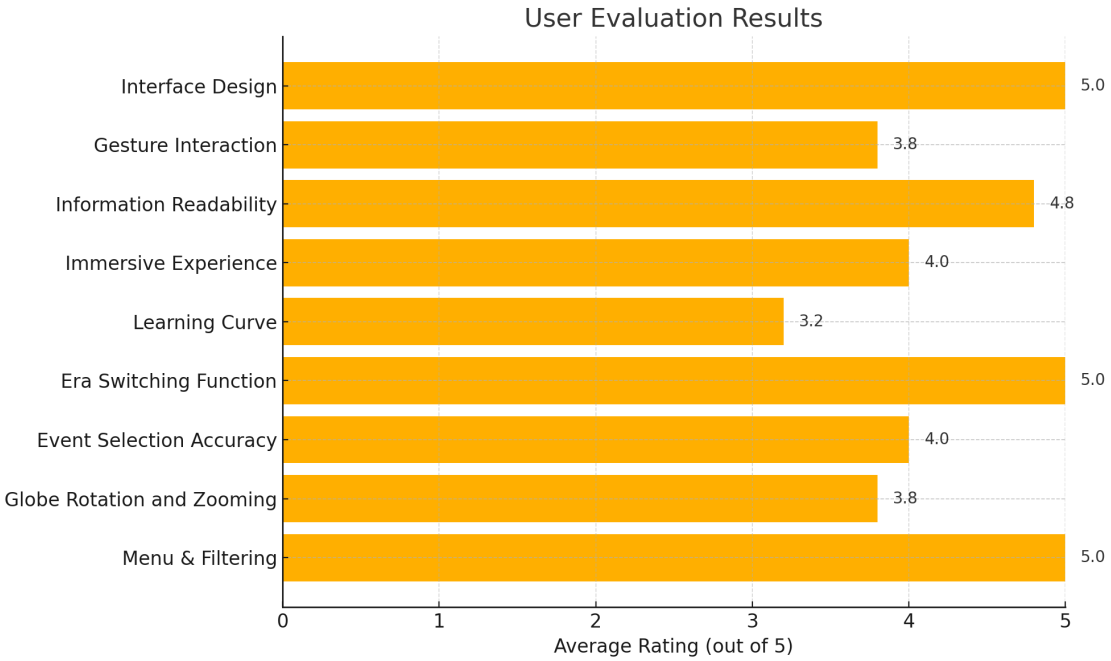

Figure 5.1: User ratings on system usability and interaction features

Key Issues Reported:

- Difficulty in Dragging: Some users, unfamiliar with HoloLens 2 gesture interactions, found it challenging to use dragging functions for the 3D globe, menu switching panel, and event display panel.

- High Learning Curve: Users generally felt that initial usage required some time to adapt to gesture controls and expressed a need for more intuitive guidance.

Desired Features:

- Voice Control: Some users expressed interest in operating the interface via voice commands to reduce reliance on gesture interactions.

Improvement Plan:

- User Guide Framework Added: The system now includes a user guide to help users familiarize themselves with basic gesture operations such as dragging, zooming, and menu switching. The guide provides interactive tutorials to assist users in gradually adapting to HoloLens 2 interactions, reducing the learning curve.

- Scene Transition Feedback Enhanced: To address the issue where users may perceive the application as frozen during scene transitions, a loading buffer animation (ProgressIndicator) has been added. Combined with asynchronous loading, this provides clear visual feedback, helping users understand that the system is still processing, thus improving the perceived responsiveness and user experience.

- Improved Dragging Experience: The sensitivity of dragging interactions for the 3D globe and UI components has been adjusted, along with enhancements to gesture recognition algorithms, improving interaction stability and fluidity.

- Exploration of Voice Control: Future versions will consider integrating voice recognition capabilities, allowing users to perform certain interactions using voice commands, thereby reducing dependence on gestures and enhancing operational convenience.

Test results indicate that the system performs well in interface design, information readability, and interaction fluidity. In response to user feedback on the learning curve, the introduction of the User Guide framework has optimized usability, and improvements to dragging interactions have enhanced operational convenience and smoothness. Future updates will further explore voice control features to meet user expectations for more intuitive interaction methods.

5.2.4 Performance Improvements

Implementation of In-System User Guide

To reduce the learning curve for first-time users, especially those unfamiliar with HoloLens gesture-based interaction, the system introduces a built-in **User Guide** interface. This interface provides a clear and accessible summary of the basic interaction methods, enabling users to engage with the 3D globe and filter historical events effectively.

The guide is implemented using the `SlateUGUI` panel framework. A `TextMeshPro (TMP)` object is embedded within the UI hierarchy `Scroll View Viewport Column2`, presenting step-by-step instructions for common tasks such as pointing, selecting, rotating, and filtering.

Figure 5.2 illustrates the final appearance of the User Guide within the HoloLens 2 environment. It describes how to use Index Finger Ray, Air Tap Gesture, Swipe and Pinch for interaction.

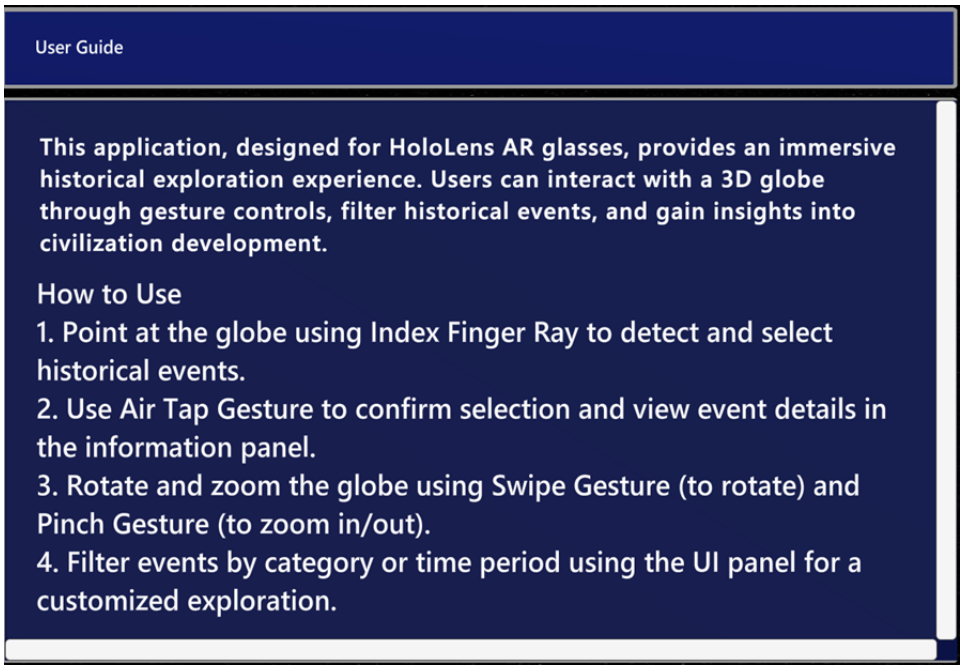

Figure 5.2: User Guide panel displayed in HoloLens 2.

This approach enhances usability by offering real-time, embedded assistance. It eliminates the need for external tutorials and reduces user frustration, especially during initial usage.

Async Scene Switching

To address user feedback regarding delays during scene switchingwhich could lead to the impression that the application is frozen or malfunctioningthe system was optimized by incorporating a **loading buffer animation** (`ProgressIndicator`) along with **asynchronous scene loading** to improve the user experience.

When users interact with the time period slider and select a different historical range, the system automatically checks whether the currently active scene matches the selected target. If not, the `LoadScene()` method is invoked to initiate the transition. This function first verifies if another scene transition is in progress to avoid overlapping calls, and then launches the coroutine `LoadSceneWithTransition()`.

The `LoadSceneWithTransition()` method encapsulates the entire scene transition logic and executes the following steps:

- Calls `progressIndicator.OpenAsync()` to display a loading animation;
- Initiates asynchronous scene loading via `SceneManager.LoadSceneAsync()` for the selected time period;

- Continuously updates the loading bar to reflect progress, providing users with real-time feedback;
- Upon completion, closes the progress indicator using `progressIndicator.CloseAsync();`
- Finally, updates the `activeScene` variable to reflect the new state and prevent redundant reloads.

The newly added code, implemented in the `SliderChangePeriod.cs` script, is shown below:

```
public void LoadScene(string sceneName)
    {
        if (sceneTransitionService.TransitionInProgress)
        {
            Debug.Log("Scene transition already in progress...");
            return;
        }
        StartCoroutine(LoadSceneWithTransition(sceneName));

        //await sceneTransitionService.DoSceneTransition(() =>
           LoadSceneAsync(sceneName));

        //StartCoroutine(LoadSceneAsync(sceneName));
    }
private IEnumerator LoadSceneWithTransition(string sceneName)
    {
        // 1. Start the progress indicator to make it visible
        if (progressIndicator != null)
        {
            Debug.Log("Opening progress indicator...");
            var task = progressIndicator.OpenAsync();
            while (!task.IsCompleted) { yield return null; }
        }
        yield return new WaitForSeconds(0.5f);

        // 2. Perform the scene transition
        yield return sceneTransitionService.DoSceneTransition(
            async () =>
            {
                Debug.Log("Loading Scene: " + sceneName);
                AsyncOperation asyncLoad = SceneManager.
                   LoadSceneAsync(sceneName);
                while (!asyncLoad.isDone)
                {
                    if (progressIndicator != null)
                    {
                        progressIndicator.Progress = Mathf.Clamp01(
                           asyncLoad.progress / 0.9f);
```

```
                    }
                    await Task.Yield();
                }
            }
        );
        yield return new WaitForSeconds(0.5f);
        // 3. Close the progress indicator
        if (progressIndicator != null)
        {
            Debug.Log("Closing progress indicator...");
            var task = progressIndicator.CloseAsync();
            while (!task.IsCompleted) { yield return null; }
        }

        // 4. Update activeScene
        activeScene = sceneName;
        Debug.Log("Scene Loaded: " + sceneName);
    }
```

Figure 5.3: Async Scene Switching

Figure 5.3 displays the loading interface shown while the scene is being loaded.

This approach enhances the systems responsiveness from the users perspective, ensuring that users are clearly informed that loading is in progressrather than mistakenly assuming the app has become unresponsive. As a result, the solution significantly improves interaction fluency and immersive experience in scene transitions.

5.3 Final Considerations

The testing and evaluation process confirmed that the system provides a stable, intuitive, and immersive user experience. Functional testing validated the correctness of key features, while performance testing demonstrated smooth interaction and efficient system responsiveness. User feedback highlighted strengths such as clear

information presentation, fluid gesture interactions, and an intuitive interface.

Despite these positive results, some areas for improvement were identified, including occasional delays in text updates, challenges in dragging interactions, and a higher initial learning curve for new users. To address these issues, enhancements such as a **User Guide framework**, optimized dragging sensitivity, and adjustments to text rendering efficiency have been implemented. Additionally, future iterations will explore the integration of voice control to further streamline interactions.

Overall, the system successfully meets its design objectives, providing an engaging and user-friendly experience on HoloLens 2. The implemented improvements, along with ongoing optimizations, will continue to enhance its usability, making it more accessible and effective for users.

Chapter 6

Discussion and Conclusion

6.1 Technical Challenges

During the development and implementation of the system, several technical challenges arose, primarily related to interaction accuracy, real-time data updating, scene transition performance, and user adaptability. Addressing these challenges required improvements in user interface design, performance optimization, and interaction techniques. This section outlines the key challenges encountered and the solutions implemented to overcome them.

6.1.1 Hand Ray Interaction with the 3D Globe

Ensuring the accuracy of hand ray interactions with the 3D globe was one of the primary challenges faced in the system. HoloLens 2 allows users to generate a ray through hand gestures, and the system needs to capture and process these interactions accurately.

- **New Hand Ray Retrieval Method:** The `TryGetHandRay(out ray)` method was introduced to obtain the ray trajectory corresponding to the user's hand gesture direction. This method utilizes HoloLens 2's hand-tracking capabilities to dynamically compute the ray's starting point and direction, allowing users to perform pointing operations more naturally.

- **Optimized Ray-Globe Intersection Calculation:** The system employs the `GetGlobeIntersection(out sphereCurrentHitPos)` method to compute the intersection between the ray and the 3D globe. This method is based on a spherical coordinate system and determines whether the ray intersects with the globe, returning the three-dimensional coordinates of the intersection point, ensuring interaction accuracy.

- **Optimized Cursor Position Update Strategy:** Once the intersection point is calculated, the `UpdateCursorLocation()` method updates the cursor position, aligning it with the ray intersection point. This enhancement provides more intuitive interaction feedback, improving the overall user experience.

6.1.2 Real-time Text Updates

During rapid event switching, the event description text may experience update delays, impacting user experience. To address this issue, the following optimizations were implemented:

- Optimized the `TextMeshPro` rendering process to improve text update speed.
- Reduced redundant computations during text content updates to enhance rendering efficiency.
- Implemented a caching mechanism to preload frequently accessed event descriptions, minimizing reload delays.

6.1.3 Scene Transition Performance

When switching between different historical periods, the system experienced noticeable loading delays, affecting the smoothness of transitions. The following optimizations were introduced:

- Loading animations were added to provide visual feedback and reduce perceived waiting times.
- Asynchronous loading techniques were implemented to preload necessary resources in the background, reducing scene transition delays.
- The memory management was optimized to prevent unnecessary resource allocations, improving overall system responsiveness.

6.1.4 User Learning Curve

Due to the novel interaction methods of HoloLens 2, some users encountered difficulties in navigation, event selection, and menu interactions. The following improvements were made to enhance user experience:

- Developed an interactive `User Guide` tutorial to provide step-by-step guidance, helping users familiarize themselves with basic operations.

- Added visual cues and tooltips to guide users through various types of interactions.
- Integrated haptic and audio feedback to enhance the intuitiveness and responsiveness of interactions.

6.2 Summary of Contributions

This research presented the design and implementation of an interactive historical visualization system using Augmented Reality (AR) on the HoloLens 2 platform. The system enables users to engage with 3D visualizations of the Earth and interact with historical data via intuitive hand gestures and ray-based selection. The key contributions and innovations are as follows:

1. **Interactive 3D Globe**

   Developed a fully interactive 3D globe optimized for HoloLens 2, supporting rotation, zooming, and position adjustment, enabling immersive spatial visualization of historical data. This feature allows users to intuitively explore historical events mapped to their geographical coordinates, enhancing spatial understanding.

2. **Fixed Event Description Display with Dynamic Updates**

   Unlike traditional cursor-following text displays, this system adopts the `SlateUGUI` component to provide a fixed event description area, ensuring a more stable text presentation. Additionally, this area dynamically updates its content based on user selections, improving readability and overall user experience.

3. **Gesture-Based 3D Interaction**

   The system leverages gesture recognition technology, allowing users to interact naturally with historical events. Gesture-based ray selection and globe manipulation enable a more intuitive and immersive exploration of historical data.

4. **Adaptive Filtering Mechanism**

   By integrating `Filter Menu Toggles`, the system enables users to filter historical events based on their interests. This feature lets users focus on specific historical periods, themes, or categories, providing a more personalized exploration experience.

5. **Enhanced Usability and Performance Optimization**

Included user guidance features and performance optimizations (e.g., asynchronous scene loading and loading animations) to enhance usability and interaction flow, providing a seamless user experience.

6. **Multi-Sensory Immersive Interaction**

   To enhance immersion, the system incorporates visual, auditory, and haptic feedback. This multi-sensory interaction enriches the user experience, making interactions more intuitive and responsive.

These contributions and innovations demonstrate the feasibility of using mixed reality technologies to facilitate immersive educational experiences. The system provides a scalable framework for visualizing spatio-temporal data in a user-centric manner, significantly improving the interactive exploration of historical content.

6.3 Limitations and Future Directions

Despite the significant progress made in interactive historical visualization, the system still has certain limitations and presents multiple potential research directions for further exploration.

1. **Adaptability of Gesture Recognition in Different Environments**

   While the system's gesture detection performs well under normal conditions, its accuracy may decrease in varying lighting conditions or when users wear gloves. Future improvements may include:

   - Integrating AI-based gesture recognition to enhance tracking robustness.
   - Introducing alternative input methods, such as eye tracking or voice control, to provide a broader range of interaction options.

2. **Scalability for Large-Scale Historical Data**

   The current system supports a limited number of historical events. Future developments could explore larger datasets while maintaining performance efficiency by:

   - Utilizing cloud-based data streaming to load historical event information dynamically.
   - Optimizing data indexing and storage structures to improve retrieval efficiency for large-scale datasets.

3. **Multi-User Collaboration Mode**

   Currently, the system is designed for single-user interaction. Expanding to a multi-user collaborative mode could enhance its application value in education and research. Potential improvements include:

- Developing real-time synchronization mechanisms to enable multiple users to explore historical data simultaneously.
- Implementing interactive annotation and discussion features to facilitate collaboration in the virtual environment.

4. **Voice Command Integration**

   Incorporating voice control as an alternative input method could improve accessibility for users with mobility limitations and enhance hands-free interaction. Potential implementations include:

   - Integrating speech recognition to trigger specific actions, such as selecting events or controlling the interface.
   - Developing customizable voice commands to accommodate diverse user preferences and enhance user engagement.

5. **AI-Driven Intelligent Recommendations**

   Future iterations of the system could incorporate artificial intelligence to provide personalized historical event recommendations based on user interactions. Possible implementations include:

   - Applying machine learning algorithms to analyze user preferences and suggest relevant historical events.
   - Developing an AI-guided assistant to help users explore historical narratives and provide in-depth insights.

This section summarized the main technical challenges that appeared during the development of the system and explained the solutions that were used to solve them. It has also outlined the system's key contributions, focusing on gesture-based interaction, visual representation of historical events, and personalized exploration functions.

Although the system achieved its primary goals, some limitations still exist. These include difficulties in recognizing hand gestures under different environmental conditions, limited capacity to handle large historical datasets, and the lack of support for multiple users interacting simultaneously. Future studies may consider using artificial intelligence to improve gesture recognition, applying cloud technologies to manage large-scale data, and developing shared interactive features for group use.

In general, the system has shown strong potential in helping users explore historical content through augmented reality. It creates an experience that is both engaging and easy to understand. The improvements suggested for the future are expected to make the system more user-friendly, expand its possible uses, and support the creation of more advanced tools for visualizing historical information in educational settings.

Bibliography

[1] Mark Billinghurst, Adrian Clark, and Gun Lee. A survey of augmented reality. *Foundations and Trendső in HumanComputer Interaction*, 8(2-3):73–272, 2015.

[2] Microsoft. Microsoft HoloLens2. `https://www.microsoft.com/en-us/hololens`. [Online]. Accessed on Mar. 20, 2025.

[3] Jorge Bacca, Silvia Baldiris, Ramon Fabregat, Sabine Graf, and Kinshuk. Augmented reality trends in education: A systematic review of research and applications. *Educational Technology & Society*, 17(4):133–149, October 2014.

[4] Nadim Mahmud, Jonah Cohen, Kleovoulos Tsourides, and Tyler M Berzin. Computer vision and augmented reality in gastrointestinal endoscopy. *Gastroenterology report*, 3(3):179184, August 2015.

[5] Microsoft. Mixed Reality Toolkit for Unity. `https://github.com/microsoft/MixedRealityToolkit-Unity`. [Online]. Accessed on Mar. 20, 2025.

[6] Andrew Barr and et al. Holoanatomy: Mixed reality anatomy teaching using microsoft hololens. *Medical Science Educator*, 30:253–260, 2020.

[7] Simone Tavella. An Educational Application of Computer Graphics for Enriching Historical Understanding of the XX Century. http://webthesis.biblio.polito.it/id/eprint/29511, 2023. Rel. Bartolomeo Montrucchio, Antonio Costantino Marceddu, Jacopo Sini.