

修士（工学）学位論文
Master's Thesis of Engineering

博物館展示を対象とした視覚トリガー型ホット
アップデート機構を用いた HMD 向け AR 展示
システム

An HMD-based AR Exhibition System for
Museum Exhibitions Utilizing a Visually
Triggered Hot-Update Mechanism

2026年3月
March 2026

閻 永祥
Yongxiang Yan



琉球大学
大学院理工学研究科
工学専攻
知能情報プログラム

Computer Science and Intelligent Systems
Engineering Course
Graduate School of Engineering and Science
University of the Ryukyus

指導教員: 赤嶺 有平
Supervisor: Yuhei Akamine

論文題目：博物館展示を対象とした視覚トリガー型ホットアップデート機構を用いた
HMD 向け AR 展示システム

氏　名：閻 永祥

本論文は、修士（工学）の学位論文として適切であると認める。

論 文 審 査 会

印
(主　査)　　赤嶺 有平

印
(副　査)　　當間 愛晃

印
(副　査)　　國田 樹

概要

本論文では、3D モデルと QR コードを統合し、HMD（ヘッドマウントディスプレイ）を通じて AR（拡張現実）空間での高度な可視化を実現するプレゼンテーションシステム「ARShowNode」と「ARShow」の開発、設計、およびその有用性について詳述する。AR 技術は現在、教育、マーケティング、娯楽など多岐にわたる分野で活用されているが、アクセシビリティとインタラクティブ性の高度な両立が依然として課題となっている。本研究では、制作者向けの「ARShowNode」プロジェクトと閲覧者向けの「ARShow」プロジェクトからなる双方向的なプラットフォームを提案し、Unity エンジンを基盤とした技術的仕様、アセットのアップロードとダウンロード、および 3D モデルの可視化プロセスについて論じる。

先行研究である「AR MUSE」などは、Android 端末を用いて 3D モデルと QR コードを紐付ける安価かつユーザーフレンドリーな解決策を提示した。これに対し、本研究の独創性は、高い没入感を実現する HMD を用いた展示において、ホットアップデートを実現した。これにより、アプリの再ビルトやストア審査を経ることなく、挙動の微調整や、展示内容の即時更新が可能となった。

システムの評価にあたっては、Unity 開発経験を有する制作者および一般の閲覧者を対象とした被験者実験を行い、システムユーザビリティの観点から定量的および定性的なアンケート実施する方針を作った。

Abstract

This paper details the development, design, and utility of "ARShowNode" and "ARShow," presentation systems that integrate 3D models with QR codes to achieve advanced visualization in Augmented Reality (AR) space via Head-Mounted Displays (HMDs). While AR technology is currently utilized in diverse fields such as education, marketing, and entertainment, achieving a high level of balance between accessibility and interactivity remains a challenge. This study proposes a bidirectional platform comprising the "ARShowNode" project for creators and the "ARShow" project for viewers, and discusses the technical specifications based on the Unity engine, asset upload and download mechanisms, and the 3D model visualization process.

Previous research, such as "AR MUSE," presented inexpensive and user-friendly solutions associating 3D models with QR codes using Android devices. In contrast, the novelty of this research lies in the implementation of a hot-update mechanism for exhibitions utilizing HMDs, which provide a highly immersive experience. This capability enables the fine-tuning of behaviors and the immediate updating of exhibition content without requiring application rebuilds or store review processes.

Regarding system evaluation, a protocol was established to conduct subject experiments targeting creators with Unity development experience and general viewers, utilizing quantitative and qualitative questionnaires to assess the system from the perspective of system usability.

研究関連論文業績

- 閻永祥, 赤嶺有平, “HMD と QR コードを活用した文化財展示向けの AR システム”
第 24 回情報科学技術フォーラム, K-003, E 棟 304, pp.495-498, 2025.

目次

第 1 章 はじめに	7
1.1 背景と目的	7
1.2 論文の構成	9
第 2 章 技術説明	10
2.1 キュレーションの変遷と定義	10
2.1.1 博物館における伝統的役割	10
2.1.2 インディペンデントキュレーターの台頭	10
2.1.3 本研究における定義と課題	11
2.2 拡張現実技術	11
2.2.1 AR の定義	11
2.2.2 ロケーションベース AR	11
2.2.3 マーカ型ビジョンベース AR	12
2.2.4 マーカレス型ビジョンベース AR	12
2.3 Unity 関連技術	13
2.3.1 Unity	13
2.3.2 Prefab	13
2.3.3 MonoBehaviour	13
2.3.4 Assembly	13
2.3.5 UUID	14
2.3.6 Asset	14
2.3.7 AssetBundle	14
2.3.8 メタファイル	14
2.4 Meta XR All-in-One SDK	15
2.4.1 パススルー機能	15
2.4.2 Meta XR Interaction SDK	15
2.4.3 Meta XR Voice SDK	15
2.4.4 WitAI	15
2.5 ホットアップデート技術	16
2.5.1 IL2CPP	16

2.5.2 HybridCLR	16
2.5.3 AddComponent メソッド	17
2.5.4 制約と課題	17
2.5.5 ホットアップデートの従来手段	18
第3章 関連研究	19
3.1 AR アプリケーションにおけるコンテンツの動的更新手法	19
3.2 クラウド統合型位置情報ベース AR コンテンツ共有システム	21
3.3 AR 展示におけるスクリーンリーダーおよび音声読み上げ技術の活用	23
3.4 博物館展示における AR インタラクション手法とデバイス特性の比較	25
3.5 XR 技術を活用した博物館および史跡における展示ガイドと地域活性化	27
3.6 ユーザの状態推定と行動ログ活用に基づく双方向的な AR 鑑賞支援	29
3.7 モバイル XR におけるリモートレンダリングと計算オフロード	31
第4章 提案システム	34
4.1 設計思想	34
4.2 開発環境	35
4.2.1 ハードウェア	35
4.2.2 ソフトウェア	38
4.3 システム構成	40
4.3.1 ARShowNode	41
4.3.2 ARShow	46
4.3.3 サーバ	48
4.4 アーキテクチャ	48
4.5 UI とインタラクション	49
4.5.1 Node0: 複合的な AR 展示インターフェース	50
4.5.2 Node1: 映像コンテンツの空間配置	52
4.5.3 Node2: 3D モデル（文化財）の展示	53
4.6 ワークフロー	54
4.6.1 制作サイド（キュレーター）	54
4.6.2 鑑賞サイド（鑑賞者）	55
第5章 評価方針	57
5.1 実験仮説	57
5.1.1 仮説 1: 制作プロセスの効率化	57
5.1.2 仮説 2: 鑑賞体験のアクセシビリティ向上	57
5.1.3 仮説 3: システム全体の統合的ユーザビリティ	58
5.2 評価手法	58
5.2.1 定量的評価（SUS）	58
5.2.2 定性的評価	58

5.3	被験者の構成	58
5.3.1	制作者 (Creator)	59
5.3.2	閲覧者 (Viewer)	59
5.4	操作説明	59
5.4.1	制作者への操作説明	60
5.4.2	閲覧者への操作説明	60
5.5	実験手順	61
5.5.1	グループ A (制作者 1、閲覧者 1 2)	61
5.5.2	グループ B (制作者 2、閲覧者 1 2)	61
第 6 章	まとめ	63
6.1	総括	63
6.2	今後の課題	64
6.2.1	評価実験の実施と検証	64
6.2.2	コンテンツの安全性審査	64
6.2.3	セキュリティの強化	64
6.2.4	クラウド環境への移行とスケーラビリティ	64
参考文献		67

図目次

1.1	ARShow システムアーキテクチャ	8
2.1	ロケーションベース AR の概念図 [5]	11
2.2	マーカ型ビジョンベース AR の概念図 [5]	12
2.3	マーカレス型ビジョンベース AR の概念図 [5]	12
2.4	Wit.AI における音声処理プロセス [5]	16
2.5	HybridCLR 構成図 [5].	17
3.1	InfoGrid システムの構成	19
3.2	MUSE システムのワークフロー	20
3.3	Clouds-Based Collaborative and Multi-Modal MR のシステムアーキテクチャ	22
3.4	ARtverse におけるサーバーからのコンテンツ読み込みと自己位置推定の プロセス	22
3.5	対話型美術鑑賞支援システムの構成図	23
3.6	AIMuseum における AR 表示例	24
3.7	Onsei AR の画面 UI	25
3.8	スマートフォンと Leap Motion を組み合わせた NI システムの構成 [8]. . .	26
3.9	クジラの骨格標本を用いた HMD 実証実験の様子 [10].	27
3.10	MuseumEye のフローティング UI とハンドジェスチャによる操作の様子 .	28
3.11	発掘調査データに基づいて復元された神崎遺跡の 3DCG	29
3.12	展示情報のリスト表示と詳細表示の UI	30
3.13	AI Aquarium のシステム概念図	31
3.14	Web ベースのリモートレンダリングシステムのアーキテクチャ	32
3.15	同期型リモートレンダリングの 3 つの状態	33
3.16	V-Light システムアーキテクチャ	33
4.1	設計思想	35
4.2	PC	35
4.3	MetaQuest3	36
4.4	USBC ケーブル	36
4.5	スマートフォン	38
4.6	MetaHorizonOSUISet	39

4.7	Meta Quest Developer Hub	40
4.8	Meta Horizon Link	40
4.9	提案システムの全体構成図	41
4.10	ARShowNodeGlobal	41
4.11	HybridCLRTTool	42
4.12	ARShowTool	42
4.13	NodeGlobal	43
4.14	EntryGlobal	44
4.15	EntryCode	44
4.16	AssetBundleGlobal	44
4.17	WitAI 配置	45
4.18	ARShowGlobal	46
4.19	ScanQr ボタン	46
4.20	Node0Progress	47
4.21	Node2Progress	47
4.22	Node0UI	47
4.23	Linkxml 配置	48
4.24	ServerTool	49
4.25	提案システムの全体構成図 [5].	49
4.26	Node0UI	50
4.27	Node0UIReader	50
4.28	Node0UIListen	50
4.29	Node0UIPoke	51
4.30	Node0UIRay	51
4.31	Node0UIGrab	51
4.32	PointableCanvasModule	51
4.33	Node0VoiceIgnoreStatus	52
4.34	Node0VoiceListenStatus	52
4.35	Node0VoiceNviCh	52
4.36	Node0VoiceNviEn	52
4.37	Node0VoicePlayStart	52
4.38	Node0VoicePlayStop	52
4.39	Node1UI	53
4.40	Node2UI	54
4.41	制作者ワークフロー	54
4.42	閲覧者ワークフロー	56
5.1	本実験の実施フロー [5].	62

表目次

4.1 開発用 PC の仕様	36
4.2 Meta Quest 3 の仕様	37
4.3 USB ケーブルの仕様	37
4.4 スマートフォンの仕様	38
4.5 開発環境およびライブラリ構成	39
4.6 開発支援ツールおよびサーバ環境	40
5.1 SUS アンケート項目	59

第1章

はじめに

1.1 背景と目的

近年、博物館や美術館をはじめとする多様な文化施設において、デジタル技術を導入した展示手法が急速に普及している。特に、AR（拡張現実）技術を用いた空間芸術展示は、物理的な制約を超えた情報の提示や、現実空間とデジタルコンテンツが融合する新たな視覚体験を提供する手段として、その重要性を増している。

こうした展示環境の変化に伴い、コンテンツの制作および運用を担う主体の役割も変容している。従来の博物館展示において、キュレーター（学芸員）の主たる役割は資料の収集・保存・調査研究であった。しかし、メディアアートやインタラクティブな展示が増加する現代においては、施設空間を活用しつつ、Unity 等のゲームエンジンを用いて自ら AR コンテンツの実装やインタラクション設計を行う、エンジニアリング能力を有したアーティストや展示製作者がその中心となりつつある。そこで本研究では、便宜上こうした「技術的背景を持ち、展示の企画から実装までを担う人物」をキュレーターと定義し、一般的な博物館法に基づく学芸員とは区別して論じるものとする。

これらエンジニアリング能力を持つキュレーターの活動は、展示表現の可能性を広げる一方で、実際の運用現場においては依然として大きな課題を抱えている。

第一の課題は、展示運営におけるコミュニケーションコストの増大である。AR 空間芸術のような専門性の高い展示では、鑑賞者が自身のデバイス等で体験を行う際、キュレーターが一人ひとりに対しアプリケーションの導入方法や操作手順を詳細に説明する必要が生じる場合が多い。鑑賞者にとっても、展示ごとに異なる独自の操作体系をその都度理解することは負担であり、これが円滑な鑑賞体験を阻害する要因となっている。

第二の課題は、コンテンツの更新性と配布プロセスの硬直性である。本研究が対象とするキュレーターは Unity 開発に習熟しているものの、iOS、Android、あるいは Meta Quest 等の HMD 向けビルドにおいては、セキュリティやプラットフォームの制約上、実行バイナリ自体の更新なしにプログラムの挙動（C# スクリプト）を変更することは極めて困難である。そのため、展示期間中に演出の微調整やロジックの修正が必要になった場合で

も、アプリケーション全体の再ビルトと再配布を余儀なくされる。特に展示会場において、Quest 端末等の HMD に対して再インストールを行う作業は、PC 接続や開発者モードでの操作を要し、1 回の更新に多大な時間を要するため、複数台のデバイスを運用する展示現場では現実的ではない。したがって、アプリ本体を更新せずに展示ロジックのみを更新可能なホットアップデート機構の確立は、展示の質と鮮度を維持するために不可欠な要素である。

本研究の目的は、上述した課題を解決するため、HMD（ヘッドマウントディスプレイ）と QR コードを活用し、制作者としてのキュレーターと閲覧者としての鑑賞者の間を媒介する中間的なアプリケーション基盤「ARShow」を提案および構築することである（図 1.1）。

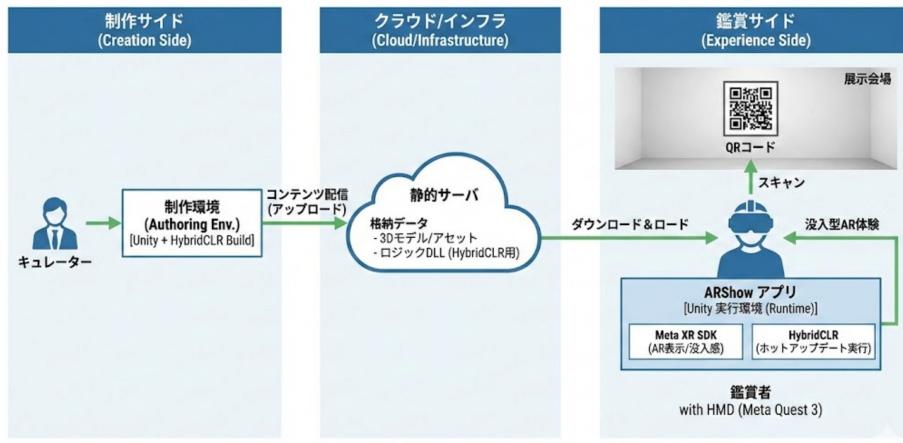


図 1.1 ARShow システムアーキテクチャ.

具体的には、以下の三つの目標を達成することを目指す。

- ・第一: Unity 向けフレームワークである HybridCLR を導入することで、従来困難であったスクリプトレベルでのコードホットアップデートを実現する。これにより、キュレーターは「ARShowTool」を通じて、自身の制作した 3D モデルや複雑なインタラクションロジックを、再ビルトの手間なく迅速かつ低コストで配信可能な環境を構築する。
- ・第二: QR コードをスキャンするだけで即座に体験を開始できる導入の簡便さに加え、HMD 特有の物理コントローラを排除したハンドトラッキングによる身体的操作や、音声コマンドによる自然言語入力を統合する。これらマルチモーダルなインタラクションの実現により、鑑賞者の学習コストを最小化し、誰もが直感的に参加できる没入感の高い AR 体験を提供する。
- ・第三: 単なる機能の提案に留まらず、Unity 上での制作、サーバへの配信、そしてクライアントアプリでの動的ロードという一連のワークフローが、実際の展示運用に耐えうるシステムとしての有効性とユーザビリティを明らかにする。

本研究を通じて、展示内容の更新がアプリケーションの再配布に依存しない柔軟な運用

環境を実現する。これにより、開発スキルを持つキュレーターにとっては表現の自由度と制作効率を高め、鑑賞者にとって手軽で没入感のある体験を提供し、新たなデジタル展示のプラットフォームを確立することを目指す。

1.2 論文の構成

- 本論文の構成は以下の通りである。
- 第2章 基礎概念を述べる。
- 第3章 関連研究を述べる。
- 第4章 提案システムを詳細に説明する。
- 第5章 評価方針を立てる。
- 第6章 総括と今後の課題を述べる。

第 2 章

技術説明

本章では、本研究が提案するシステム設計および実装の基盤となる概念と技術的背景について述べる。まず、展示の主体であるキュレーターの役割の変化と、デジタル展示における課題を整理する。次に、その解決手段としての AR（拡張現実）技術の定義と分類について概説する。続いて、実装環境である Unity および Meta XR SDK の特性を述べ、本研究の核心技術である HybridCLR を用いたホットアップデート機構、およびシステム全体の運用を支える通信アーキテクチャについて詳述する。

2.1 キュレーションの変遷と定義

2.1.1 博物館における伝統的役割

伝統的にキュレーター（学芸員）は、博物館法や ICOM（国際博物館会議）の規定に基づき、資料の収集、保存、調査研究、および展示企画を専門的に担う職種として定義されてきた。博物館という制度的枠組みの中で、歴史的かつ芸術的価値を持つ資料を体系化し、教育的配慮のもとで公衆へ提示することがその主たる役割であった。

2.1.2 インディペンデントキュレーターの台頭

近年、特定の博物館組織に所属せず、独自の文脈とテーマ設定によって展覧会を構成するインディペンデントキュレーターの活動が顕著となっている。Obrist ら [1] が指摘するように、現代のキュレーターの役割は単なる管理や保存から、新たな意味を創出するプロデューサーとしての側面を強めている。彼らの活動領域は物理空間にとどまらず、デジタル技術を用いた空間表現にも及んでおり、展示空間そのものの再定義を行っている。

2.1.3 本研究における定義と課題

AR を用いた空間芸術展示において、キュレーターの役割は鑑賞体験全体の設計者へと拡張されている。しかし、高度なデジタル技術の導入は、鑑賞者に対するデバイス操作説明やアプリケーション導入支援といった、展示の本質とは異なるコミュニケーションコストの増大を招いているのが現状である。本研究では、デジタル空間で展示構成を行い、鑑賞者へ体験を提供する主体として「キュレーター」を定義する。その上で、技術的な障壁を取り除き、彼らが表現活動に専念できる環境の構築を目指す。

2.2 拡張現実技術

2.2.1 AR の定義

AR (Augmented Reality) とは、実世界の情報にコンピュータ生成情報をリアルタイムに重畠し、人間の知覚を拡張する技術である。Azuma [1] による定義では、以下の 3 要素を満たすものとされる。

- 現実と仮想の結合 (Combines real and virtual)
- リアルタイムなインタラクション (Interactive in real time)
- 三次元的な位置合わせ (Registered in 3D)

2.2.2 ロケーションベース AR

ロケーションベース AR は、GPS (全地球測位システム) や磁気センサ、加速度センサ等の位置情報を利用し、特定の地理的座標にデジタルコンテンツを配置する手法である(図 2.1)。本手法は Pokemon GO に代表されるような広域な屋外展示には適している。しかし、屋内においては GPS 信号の遮断により位置特定精度が著しく低下することや、高さ方向の正確な整合 (レジストレーション) に課題が残る場合が多く、ミリ単位の配置精度が求められる精密な芸術作品の展示には向きである。

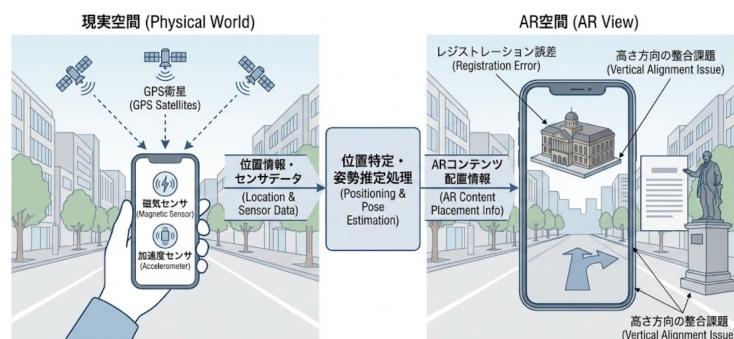


図 2.1 ロケーションベース AR の概念図 [5]

2.2.3 マーカ型ビジョンベース AR

マーカ型ビジョンベース AR は、特定の画像（マーカ）をカメラで認識し、その特徴量に基づいてデジタルコンテンツの表示位置や傾きを決定する手法である（図 2.2）。本研究では、AR コンテンツの識別子（ID）と空間的な配置基準点（空間アンカー）の両方の機能を併せ持つ QR コードをマーカとして採用する。QR コードを用いることで、画像認識の安定性が向上し、鑑賞者は意図した作品を正確な位置座標で呼び出すことが可能となる。



図 2.2 マーカ型ビジョンベース AR の概念図 [5]

2.2.4 マーカレス型ビジョンベース AR

マーカレス型ビジョンベース AR は、特定のマーカを必要とせず、SLAM（Simultaneous Localization and Mapping）技術等を用いて周囲の環境形状をリアルタイムに解析し認識する手法である（図 2.3）。本研究で使用する HMD（ヘッドマウントディスプレイ）である Meta Quest 3 等の最新デバイスでは、深度センサとカメラを用いた高度な空間認識が可能であり、壁面や床面を物理的な制約としてデジタルコンテンツに反映させることができる。しかし、この方式は環境特徴点の抽出と追跡に多大な計算リソースを要するため、モバイル HMD 単体での動作においては、マーカ型と比較してリアルタイム性および長時間稼働時の安定性に課題が残る。

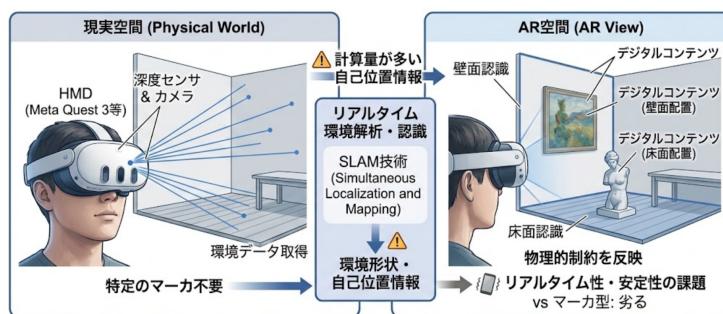


図 2.3 マーカレス型ビジョンベース AR の概念図 [5]

2.3 Unity 関連技術

2.3.1 Unity

Unity は Unity Technologies 社 [3] が提供するリアルタイム 3D 開発プラットフォームであり、現在の XR コンテンツ開発におけるデファクトスタンダードである。物理演算、レンダリング、オーディオ処理などの機能が統合されたゲームエンジンであり、C# スクリプトによる柔軟なロジック記述が可能であることから、ゲーム産業のみならず建築、自動車、学術研究など多岐にわたる分野で利用されている。

2.3.2 Prefab

Prefab は GameObject、コンポーネント（C# スクリプト）、およびプロパティ設定を一つのアセットとしてテンプレート化する機能である。これにより、展示作品を構成する複雑なオブジェクト群（3D モデル、テクスチャ、アニメーション、挙動スクリプト等）を一つの単位として管理し、実行時に動的に生成（インスタンス化）または破棄することが容易となる。

2.3.3 MonoBehaviour

MonoBehaviour は、Unity におけるすべてのスクリプトコンポーネントが継承すべき基底クラスである。本クラスを継承することで、スクリプトは GameObject にアタッチ可能なコンポーネントとして機能し、Unity のイベントライフサイクル（Start, Update, OnDestroy 等）にフックされる。本研究において、各展示作品の固有の振る舞い（アニメーション制御やインタラクション処理）はすべて MonoBehaviour を継承した C# クラスとして実装される。これにより、Unity のインスペクタ上でのパラメータ調整が可能となり、開発効率とメンテナンス性が担保される。

2.3.4 Assembly

Assembly（アセンブリ）とは、C# コードがコンパイルされた後のバイナリ単位 (.dll) を指す。Unity では通常、ユーザーが記述したスクリプトは「Assembly-CSharp.dll」という単一のアセンブリにコンパイルされる。しかし、大規模な開発や本研究のような動的な機能追加を行う場合、コードを機能ごとに分割し、独自の Assembly Definition（アセンブリ定義）を作成して管理することが推奨される。後述する HybridCLR は、このアセンブリ単位でのロードと実行制御を行うことで、スクリプトのホットアップデートを実現している。

2.3.5 UUID

Unity はプロジェクト内のアセット（ファイル）を管理するために、ファイルパスではなく、UUID（Universally Unique Identifier）、Unity 上では一般に GUID（Global Unique Identifier）と呼ばれる一意の識別子を使用する。

ファイル名やディレクトリ構造が変更された場合でも、GUID が維持されている限り、Unity はアセット間の参照関係（例えば、Prefab がどのテクスチャを使用しているか等）を正しく解決できる。本研究のホットアップデート機構においても、サーバから取得したリソースとローカルのスクリプトを正しくリンクさせるために、この一意性が重要な役割を果たす。

2.3.6 Asset

これは、Prefab 内にシリアル化されたスクリプト参照を、実行時にロードした最新のスクリプトコードへと自動的に解決（リマップ）する手法である。ただし、技術的な注意点として、スクリプトをデシリアル化する際には、生成時と同一のメタデータ構造を保持している必要がある。そのため、異なる Unity プロジェクトで生成された Prefab は、UUID の不一致等により中身のスクリプトを自動的にリマップできない場合がある。本研究では、この制約を考慮したアセット管理フローを構築している。

2.3.7 AssetBundle

AssetBundle は Unity のアセットを実行時に外部からロード可能な形式でアーカイブする機能である。本研究では、1 つの展示作品を 1 つの AssetBundle に対応させる設計を採用している。これにより、アプリケーション本体（バイナリ）を更新することなく、サーバ上のアーカイブファイルを差し替えるだけで、コンテンツの追加や更新を行うことが可能となる。

2.3.8 メタファイル

Unity プロジェクト内のすべてのアセットファイルには、対となる「.meta」ファイルが自動生成される。このメタファイルには、前述の GUID や、アセットごとのインポート設定（テクスチャの圧縮形式やモデルのスケール設定等）が記録されている。

バージョン管理システムを利用する際や、外部からアセットを取り込む際には、このメタファイルを正しく同期させる必要がある。メタファイルの欠損や不整合は参照切れ（Missing Reference）を引き起こし、アプリケーションの動作不全に直結するためである。

2.4 Meta XR All-in-One SDK

2.4.1 パススルー機能

Meta XR SDK は、Meta Quest シリーズのハードウェア機能を Unity 上で制御するための開発キットである。特に本研究では、外部カメラで取得した現実映像に CG を合成するカラーパススルー機能を活用する。これにより、現実空間と展示コンテンツがシームレスに融合した AR (Mixed Reality) 体験を構築し、鑑賞者が現実の展示会場の文脈を失うことなく作品を鑑賞できる環境を提供する。

2.4.2 Meta XR Interaction SDK

Meta XR Interaction SDK は、ハンドトラッキングやコントローラ操作を抽象化し、標準的なインタラクションを提供するライブラリである。本 SDK を用いることで、開発者はハードウェアごとの差異を意識することなく実装が可能となる。また、鑑賞者は「掴む (Grab)」「指差す (Poke)」といった直感的な身体動作で AR コンテンツを操作することが可能となり、没入感を阻害しない自然な操作体系 (NUI: Natural User Interface) が実現される。

2.4.3 Meta XR Voice SDK

Meta XR Voice SDK は音声認識モジュールであり、アプリケーションに対して音声入力インターフェースを提供する。本 SDK は、マイクから取得した音声データの正規化やストリーミング処理を担い、後述する自然言語処理プラットフォーム WitAI との通信を仲介する。これにより、HMD 装着時のハンズフリー操作や、コントローラでは表現しきれない「作品解説の呼び出し」「シーン切り替え」といった抽象度の高いコマンドを、直感的な音声入力によって実装することが可能となる。

2.4.4 WitAI

WitAI は Meta 社が提供する自然言語処理 (Natural Language Processing) プラットフォームであり、ユーザーの非構造化データ（音声やテキスト）をコンピュータが処理可能な構造化データへと変換するクラウドサービスである。本研究において、WitAI は鑑賞者の発話意図を解析し、具体的な操作命令へと変換する核心的なエンジンとして機能する。WitAI の処理プロセスは図 2.4 に示すように、主に以下の要素によって構成される。

- **Intent (インテント)** : ユーザーの発話が「何をしようとしているのか」という意図を定義したものである。例えば、「解説を再生して」や「次の作品へ移動」といった発話に対し、それぞれ「PlayDescription」や「MoveToNext」といった識別子を割り当てる。システムは返却されたインテント識別子に基づき、実行すべき C# メソッドを決定する。
- **Entity (エンティティ)** : 発話に含まれる具体的なパラメータや変数を抽出するた

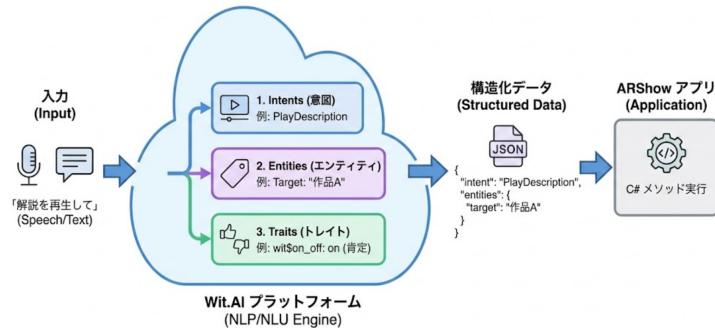


図 2.4 Wit.AI における音声処理プロセス [5]

めの定義である。例えば、「作品 A を見せて」という発話において、「見せて」はインテントであるが、「作品 A」は操作対象を特定する重要な変数である。Wit.AI は事前に学習させたキーワードや文脈に基づき、このような固有名称を抽出し、引数としてアプリケーションへ返却する。

- **Trait (トレイト)** : 発話全体の意味合いやニュアンスを分類する機能である。肯定 (Yes) または否定 (No) の判定や感情分析などに用いられる。これにより、確認ダイアログに対する応答判定などが容易になり、より自然な対話フローの構築が可能となる。

これらの機能により、Wit.AI は単なる音声のテキスト化 (Speech-to-Text) にとどまらず、文脈理解 (Natural Language Understanding) を伴う高度なインタラクションを実現する。

2.5 ホットアップデート技術

2.5.1 IL2CPP

Unity の標準的なビルド方式である IL2CPP (Intermediate Language to C++) モードでは、C# コードがビルド時に C++ へ事前訳出 (AOT: Ahead-Of-Time) される。この仕組みにより、実行速度の向上やセキュリティの確保が可能となる反面、実行中に新たな C# コード (アセンブリ) を動的に読み込んで実行することは、メモリ管理や実行権限の構造上、不可能であった。

2.5.2 HybridCLR

C# は IL (Intermediate Language) を前提とし、ホットアップデートが容易な言語として、IL2CPP の制約を受けその利点を十分に活かせていなかった。HybridCLR は、この IL2CPP 環境下において AOT 実行とインタープリタ実行を混在させることで、C# のホットアップデートを復活させたフレームワークである。図 2.5 のように示し、HybridCLR

は IL2CPP のランタイムを拡張し、サーバからダウンロードされた DLL (アセンブリ) 内のメタデータと IL 命令を直接解釈し実行するインタプリタ機能を提供する。その一方、この方式でセキュリティリスクを増大させる可能性があるため、HybridCLR は信頼できるソースからのアセンブリのみをロードすべきである。

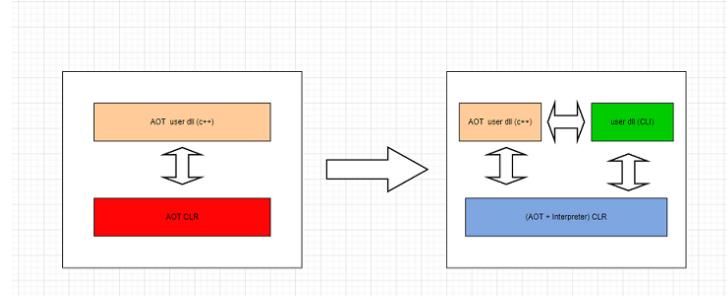


図 2.5 HybridCLR 構成図 [5].

2.5.3 AddComponent メソッド

これは、実行時（ランタイム）にロードしたアセンブリ内のクラス情報を利用し、`AddComponent` メソッドを通じて既存の `GameObject` に新たなコンポーネント（ロジック）を動的に付与する手法である。これにより、アプリリリース時には存在しなかった全く新しい機能を AR 展品に追加できる。本研究ではこの技術を採用することで、キュレーターが作成した複雑なインタラクションロジックを、`AssetBundle` として鑑賞者アプリへ即座に反映させることを可能にしている。

2.5.4 制約と課題

本システムの実装および運用において、最大の課題となるのがプラットフォーム各社が定めるアプリケーションストアの審査ガイドラインとの整合性である。Apple App Store、Google Play、および Meta Quest Store 等の主要なプラットフォームでは、セキュリティ上の理由から、アプリケーション審査完了後に外部から実行可能なコード（バイナリやスクriプト）をダウンロードし、アプリの挙動を著しく変更することを原則として禁止している。本研究で採用する HybridCLR は、コンパイル済みの DLL ファイルをサーバから取得して実行する仕組みであるため、これらの規約に抵触し、ストア審査を通過できないリスクが存在する。

この課題に対する回避策として、博物館が所有するデバイスを貸し出す形式（展示側提供環境）であれば、公式ストアを経由しない「サイドロード」や、組織内向けの「エンタープライズ配信」を用いることで、規約の制約を受けずに本システムの全機能を利用可能である。これは、特定の展示施設内で完結する運用においては十分かつ現実的な解である。

2.5.5 ホットアップデートの従来手段

アプリケーション本体（バイナリ）を再ビルドと再配布することなく、コンテンツやロジックを更新する手法自体は、ゲーム開発分野においては新しい概念ではなく、既に確立された手法がいくつか存在する。ここでは、代表的な既存手法とその課題について述べ、本研究のアプローチとの差異を明確にする。

AssetBundle によるリソース更新

Unity における最も標準的な更新手法は AssetBundle の利用である。これにより、3D モデル、テクスチャ、音声データなどの「アセット」を動的にロードすることが可能である。しかし、iOS や Meta Quest 等の IL2CPP ビルド環境下では、C# スクリプト（コンパイルされた IL コード）を AssetBundle に含めても、AOT（事前コンパイル）の制約により実行時にロードできないという技術的な制限がある。そのため、AssetBundle 単体では、展示ごとの複雑なインタラクションロジックの追加や変更には対応できない。

スクリプト言語（Lua/JavaScript）の組み込み

ロジックのホットアップデートを実現する最も一般的な手法は、Lua や JavaScript といった軽量スクリプト言語のインタプリタをアプリに内蔵する方法である。この手法では、更新対象のロジックをテキストベースのスクリプトファイルとして配信し、実行時にインタプリタがそれを解釈・実行する。これは多くの商用モバイルゲームで採用されている信頼性の高い手法である。

しかし、この手法には以下の課題が存在する。

- **開発言語の分断:** アプリ本体は C# で開発されているにもかかわらず、更新部分は Lua 等の別言語で記述する必要があり、開発者には多言語の習得が求められる。
- **バインディングコスト:** C# の関数をスクリプト言語から呼び出すためには、ラッパー（バインディングコード）の生成が必要であり、これに伴う実行時のオーバーヘッド（マーシャリングコスト）が発生する。
- **デバッグの困難さ:** C# とスクリプト言語を行き来する処理となるため、統合開発環境（IDE）でのデバッグやプロファイリングが複雑化する。

これに対し、本研究で採用する HybridCLR は、更新用プログラムも C# で記述し、Unity の API をネイティブに近い形で直接利用可能にするものである。これにより、「開発環境の統一」と「バインディングコストの排除」を実現している点が、従来のスクリプト言語方式と比較した際の最大の技術的優位性である。

参考文献 [1] Azuma, R. T. (2017). A Survey of Augmented Reality, Presence: Teleoperators and Virtual Environments6.

第3章

関連研究

3.1 AR アプリケーションにおけるコンテンツの動的更新手法

博物館や美術館における AR 展示システムの運用において、展示内容の変更や追加に合わせてアプリケーションを柔軟に更新できることは極めて重要である。Ohlei ら [1] は、博物館の専門家がプログラミングの知識なしに AR ツアーを作成および編集できるシステムとして、Ambient Learning Spaces (ALS) フレームワークの一部である InfoGrid を提案している。そのシステム構成を図 3.1 に示す。

図

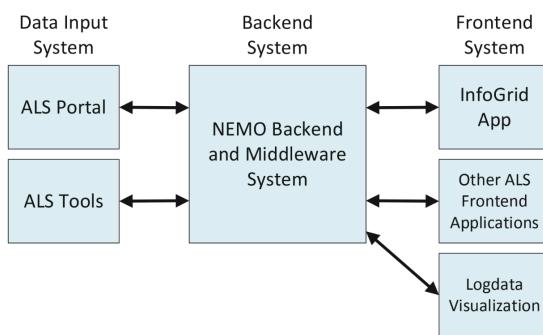


図 3.1 InfoGrid システムの構成 [1].

このシステムは、Web ベースのポータルサイト (ALS Portal) とモバイル AR アプリ、そしてバックエンドシステム (NEMO) から構成される。彼らは、静的な画像や動画だけでなく、アニメーションやインタラクティブな挙動を含む 3D オブジェクトを動的に扱うために、Asset Collections と呼ばれる新しいオーバーレイタイプを開発した。これは Unity AssetBundle 機能を活用したもので、Unity エディタ用のアドオンツールを用いて

作成されたモデルやテクスチャ、アニメーションなどのアセット群を、ランタイムでモバイルアプリにロードする仕組みである。InfoGrid アプリは、起動時に NEMO バックエンドからツアーアイテム情報を取得し、必要な Asset Collections を動的にダウンロードする。これにより、アプリ自体をアピリティア経由で更新することなく、展示物に対して動的な 3D コンテンツを追加したり、アプリ内の管理者機能で配置（移動や回転、拡大縮小）を調整したりすることが可能となる。彼らは自然史博物館における実証実験を通じて、このシステムが System Usability Scale (SUS) 86.04 という高いスコアを記録し、専門的な開発スキルを持たない博物館スタッフでも高度な AR ツアーアイテムの運用が可能であることを示している。

また、AR 展示制作のアクセシビリティ向上とコンテンツ配信の効率化に関して、Duanmu ら [2] は MUSE システムを設計および実装している。彼らの研究は、3D アーティストやキュレーターが、複雑な技術的障壁なしに自身の作品を AR 展示できることを目的としている。MUSE システムは、コンテンツ制作側の Unity Toolkit と、鑑賞者側の Viewer App からなる分離型アーキテクチャを採用しており、そのワークフローは図 3.2 の通りである。

図

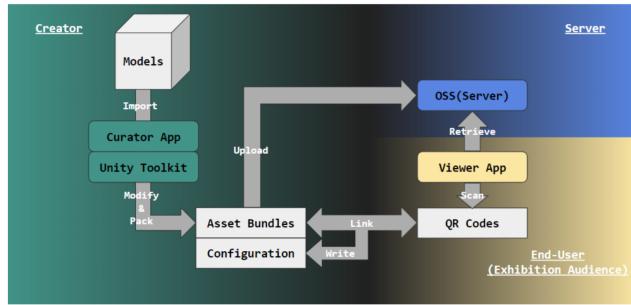


図 3.2 MUSE システムのワークフロー [2].

Unity Toolkit は Unity エディタ上で動作し、3D モデルのインポートからカスタマイズ、そして AssetBundle 形式へのパッキングまでを一貫して支援する。特に Config Generator 機能により、展示物のタイトルや説明、ファイル名などのメタデータを記述した設定ファイル (XML) を生成し、これをクラウドサーバー (Object Storage Service) にアップロードすることで展示構成を管理する。一方、Viewer App は Unity の AR Foundation を用いて開発されており、展示会場の QR コードをスキャンすることで、対応する設定ファイルを読み込み、サーバーから必要な 3D モデル (AssetBundle) をオンデマンドでダウンロードして表示する。QR コードは空間的なアンカーとしてだけでなく、コンテンツへのディープリンクとしても機能する。このアーキテクチャにより、設定ファイルを書き換えるだけで展示リストの更新やコンテンツの差し替えを行うホットフィックスが可能となり、鑑賞者用アプリの再配布を不要にしている。

これらの研究は、Unity の AssetBundle 機構とクラウド通信を組み合わせることで、アプリケーションのバイナリ更新を回避し、アセットレベルでの動的更新（ホットアップデート）を実現した成功例であると言える。次節では、こうしたコンテンツ更新の仕組み

に加え、位置情報技術と統合することで体験を共有可能にするシステムについて述べる。

3.2 クラウド統合型位置情報ベース AR コンテンツ共有システム

前節で述べたコンテンツの動的管理に加え、近年では AR (Augmented Reality) や MR (Mixed Reality) 技術において、クラウドコンピューティングや高度な位置特定技術を統合することで、単一のデバイス内で完結していた体験を、複数のユーザー間で共有かつ持続させる試みが活発に行われている。特に、特定の場所に紐づいたコンテンツをクラウド経由で管理および配信するシステムは、文化遺産の保護、ユーザー生成コンテンツ (UGC) の普及、そして芸術鑑賞の深化といった多様な領域で応用が進んでいる。

Bekele [3] は 2021 年、バーチャルヘリテージ (VH) 分野において、クラウドコンピューティングと MR 技術を統合した Clouds-Based Collaborative and Multi-Modal MR システムを提案した。従来の MR アプリケーションはデバイスの計算リソースやストレージ容量に依存するため、高精細な 3D モデルやマルチメディアコンテンツの扱いに制約があり、またアプリケーション自体の長期的な保存や維持も課題であった。そこで Bekele らは、Amazon Web Services (AWS) と Microsoft Azure という異なるクラウドプロバイダーのサービスを複合的に活用するアーキテクチャを構築した (図 3.3)。具体的には、Azure Spatial Anchors を用いて複数の HoloLens デバイス間で空間アンカー (Spatial Anchors) の識別子を共有し、Azure Cosmos DB および Azure App Service を介して同期することで、同一の現実空間におけるユーザー間の協調的なインタラクションを実現している。さらに、Amazon Polly による音声合成機能や Amazon S3 へのデータオフロードを実装することで、デバイスの負荷を軽減しつつ、文化財に関する多言語ガイドやリッチな視聴覚体験を提供可能とした。このアプローチにより、没入型技術を用いた文化学習における社会的プレゼンス (Social Presence) と仮想的プレゼンス (Virtual Presence) の両立が示された。

続いて 2023 年、Kidman [4] は、一般ユーザーが位置情報に基づくマーカレス AR コンテンツをその場で制作 (In-Situ Creation) し、共有するためのプラットフォーム ARtverse を開発した。既存の AR オーサリングツールは、QR コード等の物理マーカーに依存するか、あるいは Adobe Aero のように位置情報に紐づかない (Location-independent) コンテンツ生成に限られる傾向があり、現実世界の特定の場所に永続的なコンテンツを配置および共有する手段は限られていた。Kidman は、Unity と ARKit、そしてバックエンドに Google Firebase を採用し、クライアント・サーバー型のアプローチでこの課題解決を試みた。提案システムでは、ユーザーが配置した 3D アセットの情報だけでなく、その環境の空間特徴点マップ (ARWorldMap) をシリアル化してクラウドデータベースに保存する (図 3.4)。これにより、別のユーザーが同一地点を訪れた際、保存された空間マップを読み込むことで、マーカレスでありながら正確なコンテンツの復元 (Relocalization) を可能にしている。また、GPS の精度誤差による位置合わせの困難さを軽減するため、コンテンツ作成時のカメラ映像 (スクリーンショット) をナビゲーションの手がかりとして提示するインターフェースを導入し、ユーザー生成型 AR 体験のアクセシビリティと再現

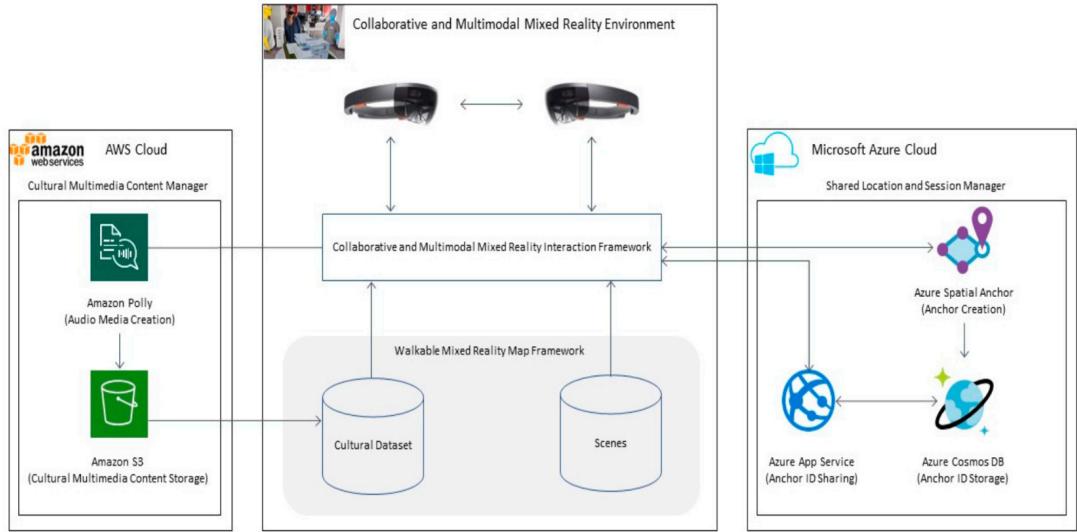


図 3.3 Clouds-Based Collaborative and Multi-Modal MR のシステムアーキテクチャ [3].

性を向上させた。

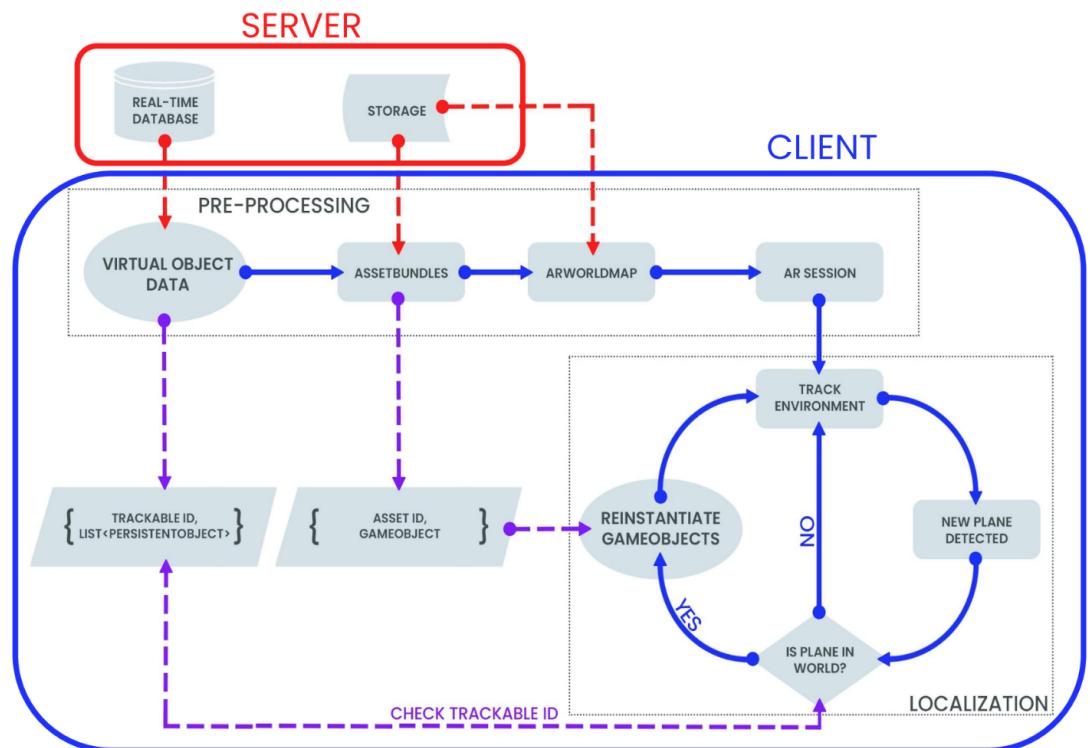


図 3.4 ARtverse におけるサーバーからのコンテンツ読み込みと自己位置推定のプロセス [4].

さらに近年、飛田ら [5] は、美術館における対話型鑑賞を支援する目的で、VPS (Visual Positioning System) を活用したマーカレス AR アプリケーションを提案している。従来の対話型鑑賞は、ファシリテータを中心としたワークショップ形式で行われることが多

く、時間的および空間的な制約により参加のハードルが高かった。彼らは、Eコマースにおける商品レビューやレコメンデーションの概念を実空間の美術作品に適用し、鑑賞体験の非同期的な共有モデルを構築した。本システムでは、鑑賞者が作品に対して宝石を模したARオブジェクトを配置し、感想コメントを紐付けることができる。これらはクラウド上で管理され、他の鑑賞者は探索モードを通じて他者の残したコメントを閲覧し、多様な視点からの鑑賞を楽しむことが可能となる。技術的には、物理的なマーカーを設置せず、VPS技術によって高精度な位置認識を行うことで、作品や展示空間の美観を損なうことなく情報の重畠を実現している（図4.1）。実証実験においては、自分の感想を即時に記録および共有できる点や、他者のコメントを通じて新たな気づきが得られる点が評価されており、ICTを活用した新たな鑑賞支援手法としての有効性が示唆されている。

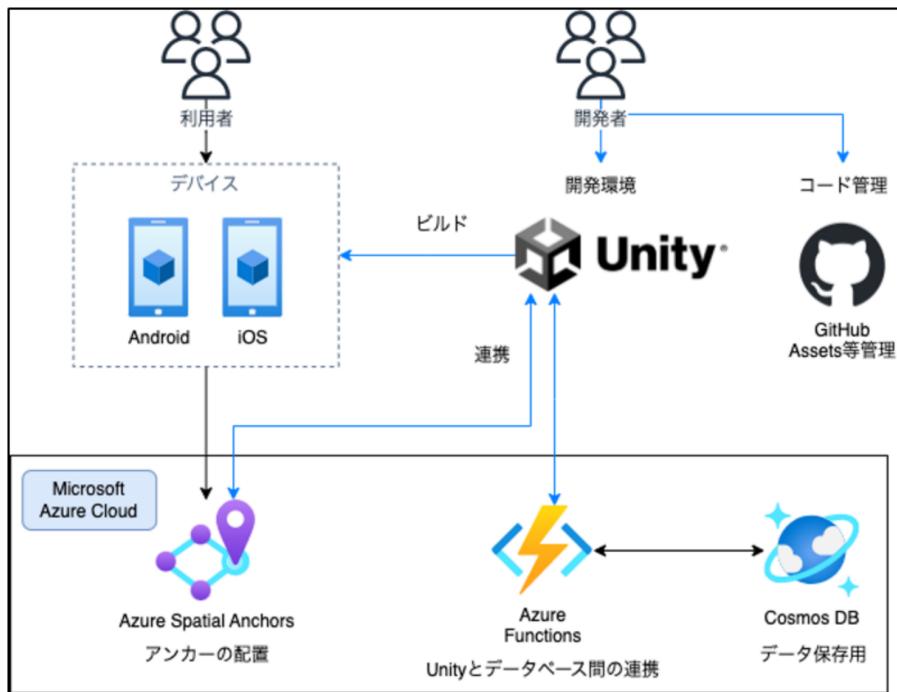


図3.5 対話型美術鑑賞支援システムの構成図 [5].

このように、クラウドと位置情報を活用したシステムは、体験の共有という側面で大きな可能性を示している。しかし、すべての来館者が視覚的な情報のみで展示を楽しめるわけではない。次節では、視覚情報に依存しないアクセシビリティ技術の統合に関する研究動向を確認する。

3.3 AR展示におけるスクリーンリーダーおよび音声読み上げ技術の活用

博物館や美術館の展示において、視覚情報に依存しない情報伝達手段として、スクリーンリーダーや音声読み上げ技術をARシステムに統合する試みが行われている。

Guedes ら [6] は、博物館におけるアクセシビリティの向上を目的として、AR 技術とスクリーンリーダーを統合したアプリケーション AIMuseum を開発した。本システムは、展示物に付与された QR コードを認識することで、関連するテキスト情報を取得し、Unity の UI Accessibility Plugin を用いて多言語で読み上げる機能を有している（図 3.6）。これにより、視覚障がい者や読字障がい者を含む多様なユーザーが、展示物の詳細情報にアクセス可能となる。38 名の参加者を対象とした評価実験では、スクリーンリーダー機能（テキスト読み上げ）に対して 92% の参加者が非常に満足と回答し、高い受容性が示された。特に、読字困難な参加者からは、スクリーンリーダーが作品への集中を助けるとの肯定的なフィードバックが得られており、視覚情報と音声情報の融合が展示体験の質を向上させることができることが確認されている。



図 3.6 AIMuseum における AR 表示例 [6].

一方、伏田と赤羽 [7] は、従来の視覚偏重型 AR の課題に対し、音声による情報提示に特化した Onsei AR を提案している。彼らは、スマートフォンの画面越しではなく、実空間の展示物を直接鑑賞させることを目的とし、画像マーカー認識と連動して解説やパネルの内容を音声で読み上げる機能を実装した。図 3.7 に示すように、アプリケーションの画面には黒背景にマーカーとの距離を示す矩形のみを表示し、視覚情報を極力排除することで、利用者の視線が手元の画面ではなく実空間の作品に向くよう設計されている。

複数の展示会における実証実験の結果、アンケートでは動画再生機能よりも音声読み上げ機能への関心が最も高く、「画面ばかり見ない体験が面白い」や「目の見えない人向けのアプリケーションとして活用できる」といった意見が寄せられた。しかしながら、展示パネルのテキストをそのまま読み上げるだけでは、単に読む労力を軽減する以上の体験価値を提供できていないという課題も浮き彫りとなり、コンテキストに応じた音声設計の重要性が示唆されている。

これらの研究は、AR 環境におけるスクリーンリーダーや音声読み上げ機能が、視覚障がい者へのアクセシビリティ支援だけでなく、晴眼者にとっても展示物への没入を阻害しない有効なインターフェースとなり得ることを示している。Guedes らのシステムはアクセシビリティの観点からスクリーンリーダーを必須機能として位置づけ、伏田らのシステムは鑑賞体験の質の観点から音声読み上げの有用性を実証しており、双方は補完的な関係にあると言える。こうした情報の提示手法に加え、ユーザーがどのようにシステムを操作するかというインタラクション手法も重要な要素である。

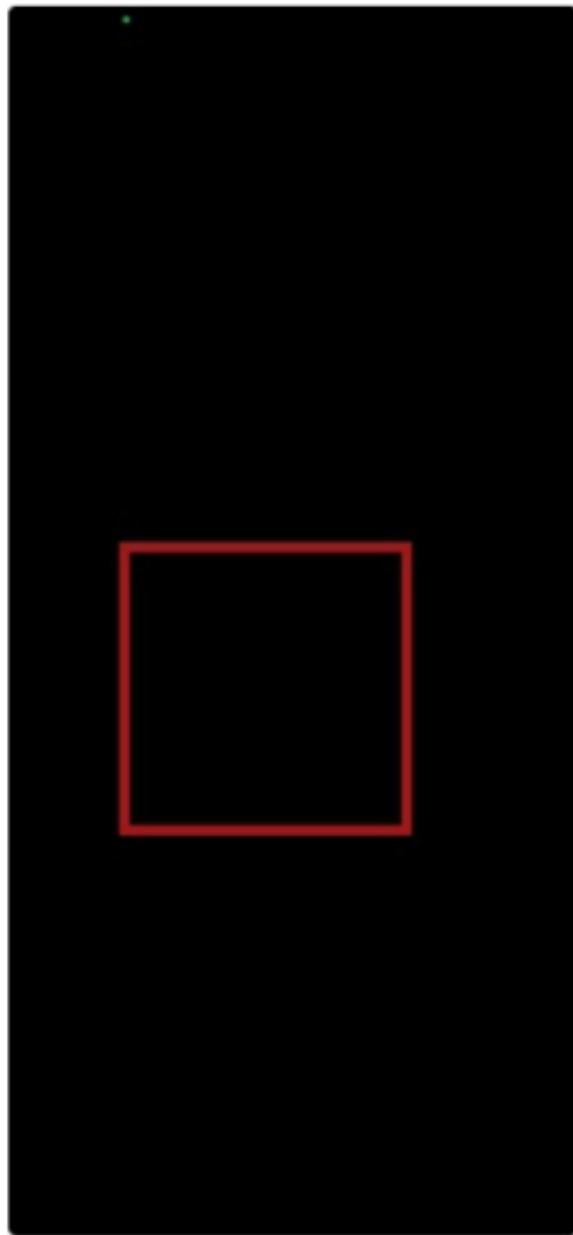


図 3.7 Onsei AR の画面 UI. 黒背景に距離を示す矩形のみを表示し、利用者が画面を注視しないよう配慮されている [7].

3.4 博物館展示における AR インタラクション手法とデバイス特性の比較

博物館や展示空間における AR (Augmented Reality) の活用において、来館者が展示物といかに直感的かつ効果的にインタラクションを行うかは重要な課題である。従来の手持ち型デバイス (Handheld Device: HHD) に加え、近年ではヘッドマウントディスプレイ (Head-Mounted Device: HMD) やジェスチャ認識を用いた手法が模索されており、それぞ

れの特性や課題に関する研究が行われている。

Kyriakou ら [8] は、博物館における展示物の接触不可という制約に着目し、Natural Interaction (NI) を用いた AR システムを提案している。彼らは、多くの博物館において展示品が保護のためにガラスケース内に収められており、来館者が物理的に触れることができないという課題に対し、スマートフォンの画面を介した従来の操作ではなく、人間の自然な手の動きを利用したインタラクションの有効性を検証した。具体的には、図 3.8 に示すように、市販のスマートフォンを HMD ケースに装着し、Leap Motion コントローラを組み合わせることで、低コストながら素手で 3D レプリカを操作（掴むことや回転させること等）できるシステムを構築した。キプロスの博物館において 60 名の来館者を対象に実証実験を行った結果、参加者の多くは当初、タッチスクリーンやボタンといった従来の操作方法を期待していたため、システムの使用に平均して約 1 分の慣れ（トレーニング）を要した。しかし、システムの使用を自然で楽しいと感じることが示され、NI が文化遺産へのアクセシビリティ向上に寄与すると結論付けている。

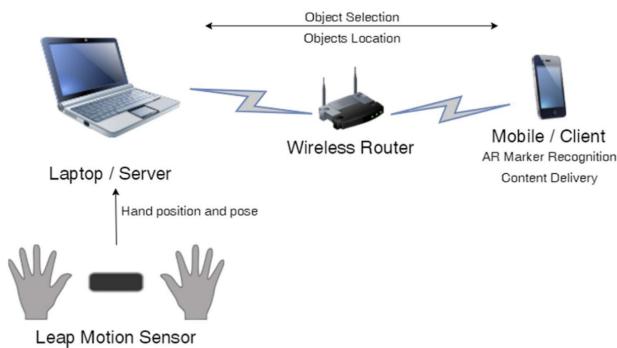


図 3.8 スマートフォンと Leap Motion を組み合わせた NI システムの構成 [8].

一方で、普及している HHD と、没入感の高い HMD との間には、インタラクションパターンの設計において明確な差異が存在する。Liu ら [9] は、博物館のガイドツアーにおける HHD と HMD (Microsoft HoloLens 2) のインタラクションパターンの比較検証を行っている。彼らは、ローマ時代の遺跡を題材とした既存の屋外用 HHD 向け AR アプリ Spirit のコンテンツを HoloLens 2 へ移植する過程を通じて、両デバイスの技術的および体験的な違いを分析した。その結果、HHD で一般的である画面タッチや GPS による位置情報トリガーといった手法は、HMD におけるエアタップ (Air Tap) や SLAM (Simultaneous Localization and Mapping) といった技術と単純に置換できるものではないことが明らかになった。特に、HMD では HHD のように画面上のテキストを読む体験が快適ではない点や、HoloLens 2 には GPS が内蔵されていないため屋外での広域ナビゲーションが困難である点などの課題が浮き彫りとなった。これらを踏まえ、彼らはデバイスの特性に応じたインタラクションパターンの再設計が必要であると論じている。

さらに、HMD を用いた AR アプリケーションを初めて利用する来館者が直面する課題について、Liu ら [10] は自然史博物館におけるクジラの骨格展示を用いたプロトタイプ開発とユーザビリティ評価を行っている。彼らは、HMD 特有の操作方法（ハンドメ

ニューの呼び出しや視線入力など) が、未経験者にとって高いハードルとなることを指摘し、オンボーディング(導入指導)、データ収集、フィードバックの3段階からなる評価手法を提案した。HoloLens 2 を用いた実証実験(図3.9)の結果、参加者はアバターへの追従や展示情報の探索といったタスクを概ね遂行できたものの、一度学習したジェスチャ操作を忘れてしまう問題や、デバイスの視野角(Field of View: FOV)の制限によりホログラムが見切れてしまうといったハードウェア由来の課題が確認された。また、評価者が横について安全を確保しつつ観察を行う手法の有効性が示され、HMD を用いた展示体験においては、初心者が操作学習に費やす時間を最小限に抑え、展示内容そのものに集中できるようなインタラクション設計と導入支援が不可欠であることが示唆されている。



図3.9 クジラの骨格標本を用いた HMD 実証実験の様子 [10].

3.5 XR 技術を活用した博物館および史跡における展示ガイドと地域活性化

博物館や史跡における展示体験の向上を目的として、MR (Mixed Reality) や VR (Virtual Reality)、AR (Augmented Reality) といった XR 技術の導入が進められている。これらの技術は、従来の物理的な展示と人間によるガイドの制約を補完し、来館者のエンゲージメントを高める手法として注目されている。

Hammady ら [11] は、博物館における従来のガイドサービスの役割を再定義し、MR 技術を用いたホログラフィック・バーチャルガイドシステム MuseumEye を提案している。彼らは、エジプト考古学博物館におけるツタンカーメン展示を対象に、Microsoft HoloLens を用いたシステムを開発した。本システムは、3D スキャンされた展示品やテキスト、画像などのマルチメディア情報を空間上に配置し、歴史的なアバター(バーチャルガイド)が来館者にストーリーテリングを行うものである。技術的な特徴として、MuseumEye は 5 つのインタラクションレベル(物理環境、物理的展示物、バーチャルガイド、バーチャル展示物、UI)を提供し、図3.10 に示すように、ユーザーはハンドジェスチャ(エアタップ)を用いて空間上のフローティング UI を直感的に操作を行うことができる。また、展示空間にゲーム性を取り入れた Knowledge Scale game を導入することで、来館者の能動的な学習を促進している。171 名の参加者を対象とした評価実験の結果、ガイド役(Role of Guide)という構成概念が、システムの有用性(Usefulness)や使いやすさ(Ease of Use)と、将来の利用意図(Intention to Use)との関係を媒介する重要な要因であることが明らかになった。これは、MR ガイドが単なる情報提示ツールではなく、人

間のガイドと同様に Pathfinder（先導者）や Mentor（助言者）としての役割を果たすことで、来館者の体験価値を向上させることを示唆している。一方で、HoloLens の視野角 (FOV) の狭さや、装着の不快感といったハードウェアに起因する課題も報告されている。

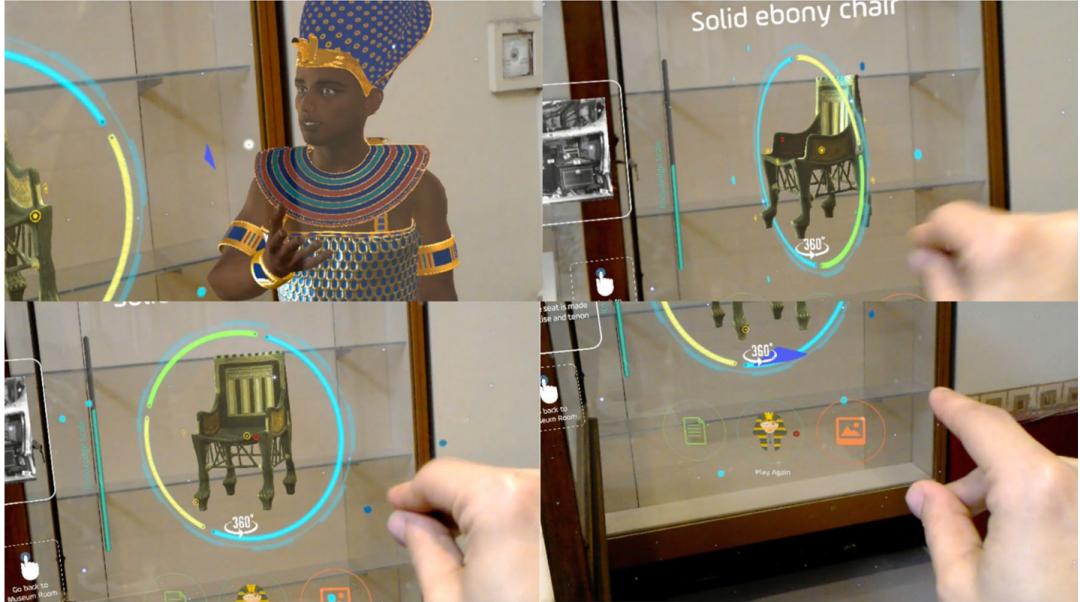


図 3.10 MuseumEye のフローティング UI とハンドジェスチャによる操作の様子 [11].

また、地域固有の文化財に対する理解促進と地域活性化（シビックプライドの醸成）を目的とした XR コンテンツの活用事例として、井上と長澤による研究 [12] が挙げられる。彼らは、神奈川県綾瀬市にある国指定史跡神崎遺跡を対象に、VR および AR コンテンツを制作し、その効果を検証している。神崎遺跡のような埋蔵文化財は、遺構が埋め戻されているため視覚的なイメージを掴みにくいという課題があった。これに対し、井上らは発掘調査データに基づき、図 3.11 のように環濠や竪穴住居を 3DCG で復元した。VR コンテンツ VR 神崎遺跡では、HTC Vive Pro を使用し、あえて自由移動ではなくカメラが自動で移動する方式を採用することで、VR 酔いの防止と操作に不慣れな利用者への配慮を行っている。さらに、現地での体験を強化するために開発された AR 神崎遺跡は、Unity と AR Foundation を用いて制作されたモバイルアプリケーションである。これは、史跡内の解説看板に設置された QR コードを読み取ることで、スマートフォンのカメラ映像に当時の集落の様子（環濠や住居）を重畠表示させるものである。これらのコンテンツを用いた実証実験の結果、参加者からは文化財のスケール感や位置関係の理解が深まったとの評価が得られ、地域住民のシビックプライド向上や観光資源としての魅力向上に寄与する可能性が示された。

以上の先行研究から、MR や AR を用いたガイドシステムは、展示物の視覚的な復元や情報付加により、来館者の理解と関与を深める上で有効であることが示されている。しかし、デバイスの視野角制限や操作性の学習コスト、あるいは現地への導入における運用上の課題など、解決すべき点も残されている。次節では、こうした一方的な情報提示を超



図 3.11 発掘調査データに基づいて復元された神崎遺跡の 3DCG [12].

え、ユーザーの行動や状態をシステムが理解し活用する双方向的なアプローチについて述べる。

3.6 ユーザの状態推定と行動ログ活用に基づく双方向的な AR 鑑賞支援

博物館や展示施設における AR (拡張現実) 技術の活用は、単にデジタルコンテンツを一方的に展示空間へ重畠表示する段階を超え、ユーザの鑑賞行動や生体反応をシステム側が取得および分析することで、より質の高い双方向的な鑑賞体験を提供するフェーズへと移行している。

Hou ら [13] は、Microsoft HoloLens を用いた博物館向けの AR 鑑賞アプリケーションにおいて、ユーザのインタラクション情報をシステムが収集し、それを展示表現に還元する仕組みを提案している。本システムでは、展示物に対する解説テキストや 3D モデルの表示といった基本機能に加え、視線入力 (Gaze) やジェスチャ、音声コマンドといったマルチモーダルな入力をサポートしている。特筆すべきは、将来のデータ分析のためにユーザの仮想情報に対する操作を記録するタスクが設計されている点である。具体的には、図 3.12 に示すように、先行する来館者のインタラクション履歴（クリック数）をシステムが蓄積し、そのデータに基づいて展示タイトルの明るさ (Lightness) を変化させる機能が実装されている。図中のリスト項目右側に付記された数値は過去のユーザによる参照回数を示しており、システムはこの数値に基づいてユーザからどの展示項目が注目されているかという情報を可視化している。これを視覚的なポピュラリティとして後続のユーザへフィードバックすることで、静的な情報提示に留まらない動的な展示空間を構築している。

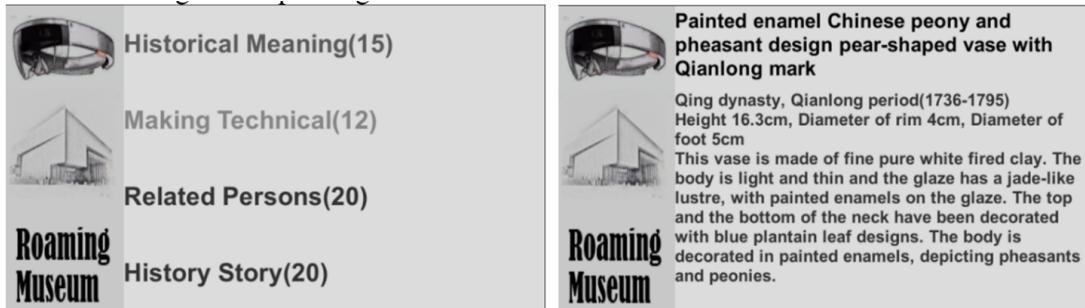


図 3.12 展示情報のリスト表示（左）と詳細表示（右）の UI [13].

モバイル端末を用いたアプローチにおいて、赤嶺ら [14] は、展示者側からの情報提供のみならず、来館者の個々の展示物に対する興味度や反応を推定することを目的とした双方向メディア型ガイダンスシステムを開発した。この研究では、タブレットやスマートフォン上で動作する AR 技術を用いながら、鑑賞者の動線や興味を推定する手法を組み込んでいる。システムはユーザの滞在時間や操作履歴といった行動データを取得および解析することで、ユーザが何に関心を持っているかを判断し、その興味に関連する類似の展示物を自動的に検索かつ提示する機能を備えている。これは、AR システムがユーザの状態を受動的に観測するだけでなく、取得した情報を能動的に活用して、個々のユーザに最適化された学習支援を行おうとする試みであると言える。

さらに近年では、高度なセンシング技術を有する HMD (ヘッドマウントディスプレイ) を活用し、より詳細なユーザ情報を取得および活用する研究が進められている。星野 [15] は、Meta Quest 3 を用いた AR 型遠隔学習支援システム AI Aquarium において、デバイスに搭載された環境察知カメラや IMU (慣性計測装置) を用いて、ユーザの視点や頭の動き、位置情報を高精度に測定する手法を採用了。本システムの全体像を図 3.13 に示す。図中の概念図にあるように、本システムではこれらのセンシングデータに基づいて鑑賞視点の同期を図るだけでなく、閲覧ログ情報をユーザの興味関心を分析するための基礎情報

として蓄積している。さらに、蓄積されたログデータと、オントロジーサーバによって構造化された知識情報を組み合わせることで、閲覧者の属性や過去の興味傾向に基づいた最適なコンテンツを提案する機能（閲覧者属性によるコンテンツ提案機能および閲覧ログによる適正コンテンツ表示機能）の実装が試みられている。これにより、システムはユーザの無意識的な鑑賞行動からも情報を取得し、対話的かつ適応的な学習環境の形成を実現している。

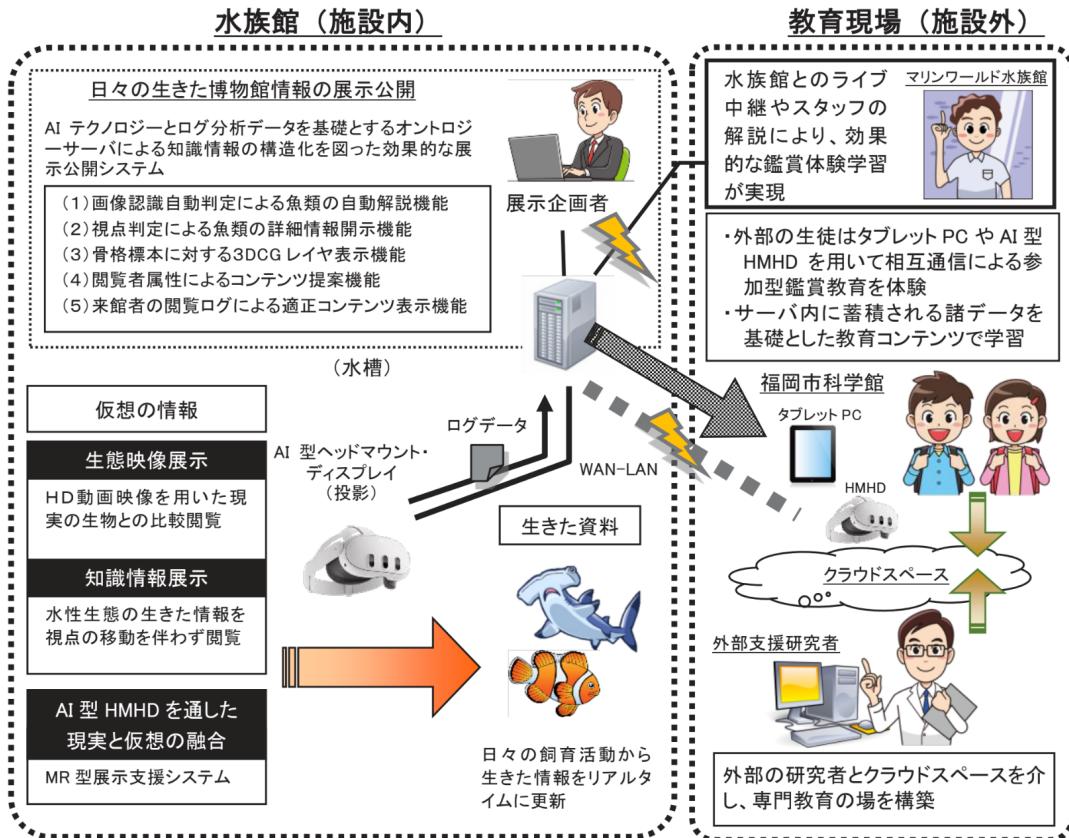


図 3.13 AI Aquarium のシステム概念図 [15].

以上のように、近年の AR 鑑賞支援システムにおいては、展示情報の提示という出力の側面だけでなく、視線や動作、操作ログといったユーザ情報の入力と解析が重要な要素となっており、それらを活用した双方向的なインタラクションデザインが研究の潮流となっている。最後に、これらの高度な AR/VR 体験をモバイル環境で実現するための技術的課題である描画処理と計算オフロードに関する研究を紹介する。

3.7 モバイル XR におけるリモートレンダリングと計算オフロード

モバイル VR デバイスにおける描画性能の制約と、クロスプラットフォーム開発のコスト削減を目的として、Seligmann [16] は、Web ベースの VR クライアントを用いたリモー

トレンダリングシステムを提案している。従来の VR アプリケーションはデバイスごとに異なる API や SDK への対応が必要であり、開発コストが増大するという課題があった。これに対し、Seligmann は Web ブラウザ上で動作するフレームワークである A-Frame と、WebRTC によるリアルタイム通信を組み合わせることで、特定のハードウェアに依存しない汎用的なクライアントシステムを構築した（図 3.14 参照）。提案システムでは、Unity で構築されたサーバー側でレンダリング処理を行い、その結果を映像ストリームとしてクライアントへ送信する。また、レイテンシの影響を軽減するために、サーバー側で生成したカスタムキューブマップを用いる手法を採用した。評価実験では Oculus Quest をクライアントとして用いたが、動画のデコード処理やシェーダーの負荷により、滑らかな再生に必要とされる 60 fps を維持することが困難であり、デコード処理の最適化が課題として残された。

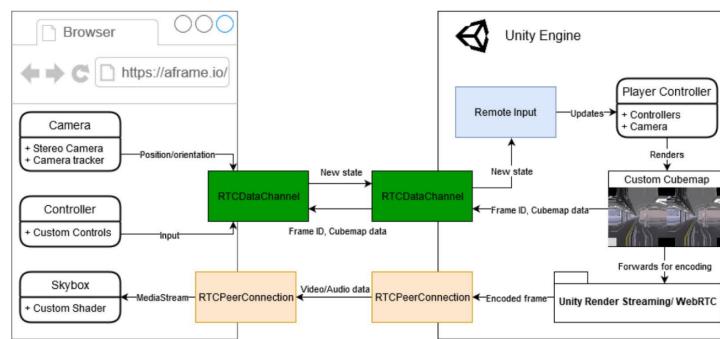


図 3.14 Web ベースのリモートレンダリングシステムのアーキテクチャ概要 [16]。

リモートレンダリングにおけるフレームタイミングの不一致とそれに伴う遅延の問題に対し、Kelkkanen [17] は、サーバーとクライアント間で厳密な同期を取る同期型リモートレンダリング (Synchronous Remote Rendering) アーキテクチャを提案している（図 3.15 参照）。従来非同期で行われていたレンダリングと表示のプロセスを同期させることで、フレームドロップやティアリングの発生を防ぎ、ネットワーク条件が良好な場合にはローカルレンダリングと同等の低遅延 (Local Latency Mode) を実現することを目指した。彼らは NVIDIA の GPU (GTX Titan X 等) が持つハードウェアエンコーダ (NVENC) を活用し、信頼性 UDP (RUDP) を用いた独自の通信プロトコルを実装した。実験では、LAN 環境および 50 km 離れたサーバーとのインターネット環境、5G ネットワーク環境下で評価を行い、有線 LAN 接続においては HTC Vive のネイティブ解像度と 90 fps のフレームレートで安定した動作を確認した。一方で、5G や不安定なネットワーク環境下では、遅延注入 (Delay Injection) や解像度の動的変更を行うことで、フレームレートの維持を図る必要性が示された。

さらに近年では、レンダリングだけでなく、計算負荷の高い認識処理をエッジサーバーにオフロードするエッジネイティブなアプリケーションも提案されている。Hammad ら [18] は、モバイル AR ゲームにおける身体動作認識の負荷をエッジコンピューティングで解決するシステム V-Light を開発した（図 3.16 参照）。モバイルデバイス単体では困難な、高精度な多人数姿勢推定 (OpenPose) をエッジサーバー上で実行し、その結果を用いてリ

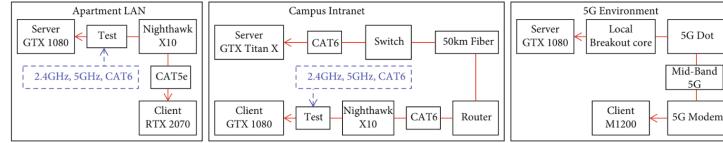


図 3.15 同期型リモートレンダリングシステムにおける 3 つの状態 [17].

アルタイムな AR シューティングゲームを実現した。V-Light のシステムアーキテクチャは、画像処理パイプラインとゲームプレイパイプラインを分離し、画像データは Unity から Gabriel クライアントを通じてサーバーへ送信され、処理された姿勢データのみがクライアントへ返送される仕組みとなっている。これにより、帯域幅の節約と応答性の向上を図っている。彼らはこのシステムを通じて、計算負荷の高い処理をエッジへオフロードすることで、従来のモバイル AR では実現不可能であった身体性を伴う同期型マルチプレイ体験が可能になることを示した。

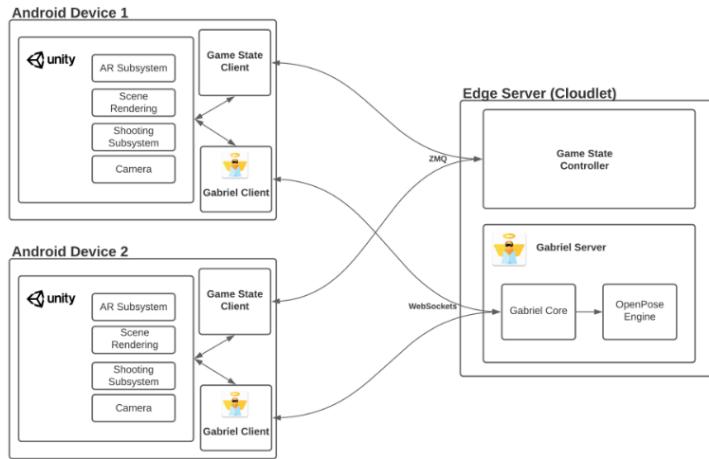


図 3.16 エッジコンピューティングを活用した V-Light のシステムアーキテクチャ [18].

第4章

提案システム

4.1 設計思想

本節では、本研究で提案した AR 展示システムの設計思想について述べる。

本システムは、AR（拡張現実）を用いた芸術展示において、キュレーター（制作者）と鑑賞者（閲覧者）の間に存在するコミュニケーションコストの削減、および展示コンテンツの持続可能な運用を解決することを主たる目的として設計された。従来のデジタル芸術展示、特にアプリベースの AR 展示においては、策展者が作品を公開するためにアプリケーションそのものを配布する必要があり、更新のたびに再配布や再インストールが求められるという課題があった。また、展示会終了とともにアプリのサポートが終了し、作品の鑑賞が不可能になるケースも散見される。さらに、鑑賞者にとっても、展示ごとに異なる操作方法や導入手順を学習する必要があり、これが鑑賞体験への障壁となっていた。

これらの課題に対し、本システムはキュレーターと鑑賞者を媒介するプラットフォームとしての役割を果たす。本システムの設計思想を図 4.1 に示す。具体的には、Unity の AssetBundle 機能と HybridCLR によるホットアップデート技術を組み合わせることで、アプリケーション本体を更新することなく、展示コンテンツ（ロジックを含む）をサーバ経由で動的に追加や更新可能なアーキテクチャを採用した。策展者は、提供されるツールキットに従い AR 展品（Node）を作成し、サーバへアップロードするだけで、恒久的なアクセスが可能となる。一方、鑑賞者は単一のクライアントアプリ（ARShow）を使用し、会場の QR コードをスキャンするだけで、即座に該当する作品コンテンツをダウンロードし、鑑賞を開始することができる。この設計により、ハードウェアやアプリの更新に依存しない作品の永続性と、直感的かつ統一された鑑賞体験の両立を目指している。

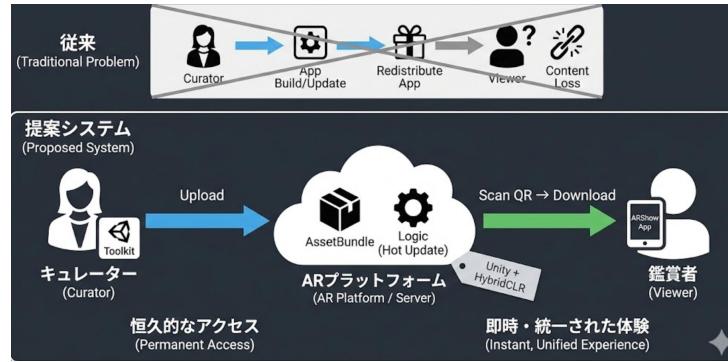


図 4.1 設計思想 [5].

4.2 開発環境

本節では、本システムの構築および動作検証に使用したハードウェアおよびソフトウェア環境について述べる。

4.2.1 ハードウェア

本システムの開発および実行には、コンテンツ制作とサーバホスティングを行う PC と、AR 体験を提供する HMD (Head Mounted Display) を用いた。

開発用 PC (図 4.2) には、Windows 11 Professional (25H2) を搭載したワークステーションを使用した。Windows 環境を採用した主な理由は、Meta Quest Link 機能を利用することで、実機へのビルトしてインストールを行うことなく、Unity エディタ上で直接 AR 空間の動作確認が可能となり、開発効率が著しく向上するためである。



図 4.2 開発に使用した PC の外観

本研究で使用した PC の具体的なハードウェア構成を表 4.1 に示す。

表 4.1 開発用 PC (Redmi G Pro 2024) のハードウェア構成

分類	項目	仕様
General	Model Name	Redmi G Pro 2024
System	OS	Windows 11 Professional
Hardware	CPU	Intel Core i7-14650HX
Hardware	GPU	NVIDIA GeForce RTX 4060 Laptop
Hardware	Memory (RAM)	32 GB (DDR5)
Hardware	Storage	1 TB SSD (PCIe 4.0)

鑑賞用デバイスには、ビデオシースルー機能を備えたスタンダードアロン型 HMD である Meta Quest 3 (図 4.3) を採用した。また、開発 PC 機と HMD の接続には、大容量データの高速転送および安定したストリーミングを実現するため、転送速度 2.5Gbps 以上の帯域を持つ USB-C ケーブル (図 4.4) を使用した。



図 4.3 MetaQuest3 [5].



図 4.4 USBC ケーブル [5].

ここで、Meta Quest 3 の主要仕様を表 4.2 にまとめた。

表 4.2 実験に使用した HMD (Meta Quest 3) の主な仕様

分類	項目	仕様
General	Model	Meta Quest 3
Processor	SoC	Snapdragon XR2 Gen 2
Display	Resolution	2064 × 2208 pixels per eye
Display	Refresh Rate	90 Hz / 120 Hz
Optics	Field of View	110° (H) / 96° (V)
Memory	RAM	8 GB
Camera	Passthrough	Full-color (4 MP, 18 PPD)

本研究で言及した USB-C ケーブルの主要仕様を表 4.3 にまとめる。

表 4.3 HMD と PC の接続に使用した USB ケーブルの仕様

分類	項目	仕様
Interface	Connector Type	USB Type-C to Type-C
Standard	USB Standard	USB 3.2 Gen 1
Performance	Max Transfer Rate	5 Gbps
Physical	Length	5.0 m
Material	Core Type	Optical Fiber

ネットワーク環境として、本研究では LAN (構内通信網) 内での運用を想定し、スマートフォン (図 4.5) のテザリング機能を用いて PC と HMD を同一ネットワークに接続した。なお、スマートフォン再起動等による IP アドレスの変更に対応するため、開発機側で固定 IP 設定もしくは動的 IP の確認手順を確立して運用した。



図 4.5 スマートフォン [5].

本研究で使用したスマートフォンの主な仕様を表 4.4 に示す。

表 4.4 使用したスマートフォンの仕様一覧

分類	項目	仕様
Device	Model Name	Google Pixel 7a
System	OS Version	Android 13
Network	Wi-Fi Standard	IEEE 802.11ax (Wi-Fi 6E)
Network	Tethering Band	5 GHz / 2.4 GHz
Hardware	Processor (SoC)	Google Tensor G2
Hardware	Memory (RAM)	8 GB

4.2.2 ソフトウェア

本システムのソフトウェア開発環境は、Unity を統合開発環境の中核とし、Meta 社が提供する XR 開発キットおよびサードパーティ製のライブラリ群によって構成されている。

主要な開発プラットフォームとして Unity を使用し、Meta Quest 3 のパススルー機能や空間認識機能を利用するため Meta XR All-in-One SDK および OpenXR プラグインを導入した。これにより、高精度なパススルー機能や空間アンカー、ハンドトラッキングといった固有機能へのアクセスを可能にしている。同時に、標準規格である OpenXR Plugin を併用することで、将来的なデバイス互換性と拡張性を担保した。ユーザーインターフェース (UI) の構築においては、Meta Horizon OS UI Set (図 4.6) を採用した。Quest のシステム標準 UI と親和性の高いデザインコンポーネントを使用することで、ユー

ユーザーに対して違和感のない、直感的な操作体験を提供することを目的としている。

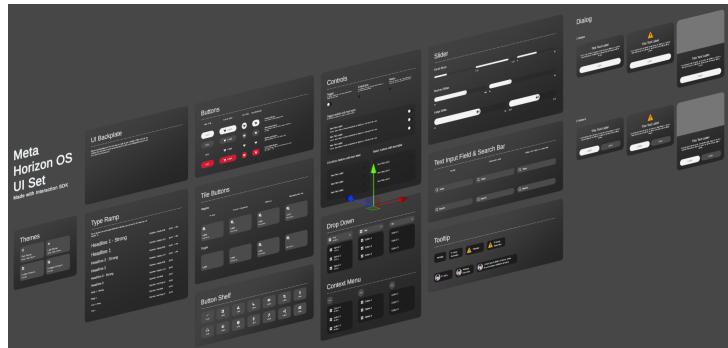


図 4.6 MetaHorizonOSUISet [5].

また、本システムのアーキテクチャ上の特徴として、動的なコード実行（ホットアップデート）機能の実装が挙げられる。通常の Unity IL2CPP ビルドでは困難な、実行時のロジック更新を実現するため、C# インタープリタフレームワークである HybridCLR を組み込んだ。これにより、実機への再インストールを行うことなく機能の拡張が可能となり、開発サイクルの大幅な効率化を実現している。その他、外部入力機能として、QR コード認識には軽量かつ高速な ZXing.unity を、音声コマンド認識には Meta Wit.ai を活用した。

本システムの開発および実装に使用した主要なプラグインとライブラリ一覧を表 4.5 に示す。

表 4.5 本システムで使用した開発環境およびライブラリのバージョン一覧

分類	名称	バージョン
Platform	Unity	6000.0.62f1
XR Plugin	Meta XR All-in-One SDK	v81
XR Plugin	OpenXR Plugin	v1.1.54
Library	HybridCLR	v8.5.0
Library	ZXing.unity	v3.5.4

実機へのデプロイ、パフォーマンス監視、および画面収録などのデバイス管理プロセスには、公式の統合開発ツールである Meta Quest Developer Hub（図 4.7）を使用した。また、PC 側でのレンダリング検証や Unity エディタの Play Mode を活用した迅速なデバッグを行うため、Meta Horizon Link（図 4.8）による有線接続環境を構築した。

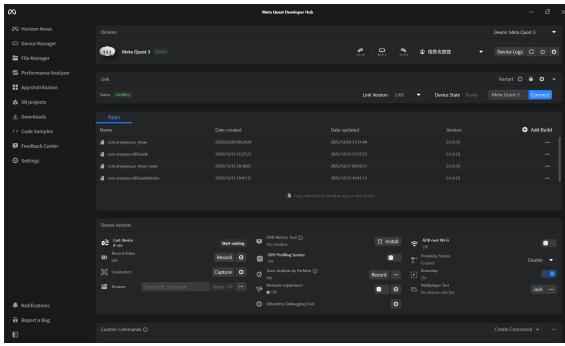


図 4.7 Meta Quest Developer Hub [5].

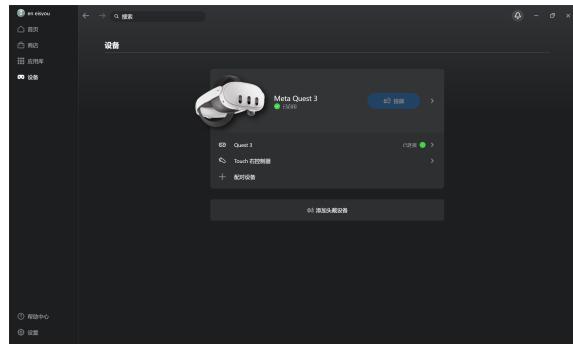


図 4.8 Meta Horizon Link [5].

サーバサイドの実装においては、クライアントサイド(Unity)と主要開発言語を C# に統一することで、データモデルの共有やコンテキストスイッチの低減を図るために、ASP.NET Core を採用した。特に、軽量かつ高速なレスポンスが求められるため、Minimal API 構成で実装している。

コーディングおよびデバッグ環境には、軽量かつ拡張性に優れた Visual Studio Code を選定した。Microsoft 社が提供する C# Dev Kit および Unity 拡張機能を導入することで、コード補完やブレークポイントによるデバッグ効率を最大化した。

本システムの開発および実装に使用した主要なソフトウェア開発環境一覧を表 4.6 に示す。

表 4.6 デバッグツール、サーバフレームワークおよびエディタのバージョン一覧

分類	名称	バージョン
Tool	Meta Quest Developer Hub	v3.2
Tool	Meta Horizon Link	v83.0.0.224.349
Framework	ASP.NET Core	v10.0.1
Editor	Visual Studio Code	v1.108

4.3 システム構成

本節では、提案システムの具体的な構成要素と、それらが連携するアーキテクチャについて詳細に述べる。提案システムは大きく分けて、

- ・キュレーターが展示コンテンツを制作および出力するための Unity プロジェクト「ARShowNode」
- ・鑑賞者が使用する閲覧用アプリケーション「ARShow」
- ・コンテンツを保持かつ配信する「静的ファイルサーバ」

の三つ要素から構成される。提案システムの全体構成を図 4.9 に示す。

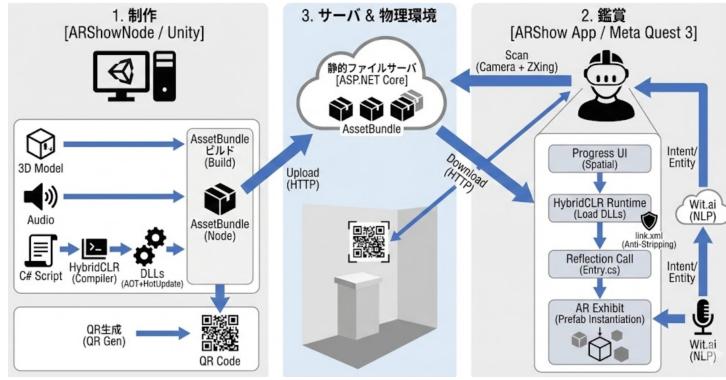


図 4.9 提案システムの全体構成図 [5].

4.3.1 ARShowNode

ARShowNode は、キュレーター（制作者）向けに提供される Unity プロジェクトである（図 4.10）。本プロジェクトは、AR 展品（Node）を単位（AssetBundle）として管理し、各 Node には 3D モデル、音声、映像、および動的ロジック（C# スクリプト）が含まれる。

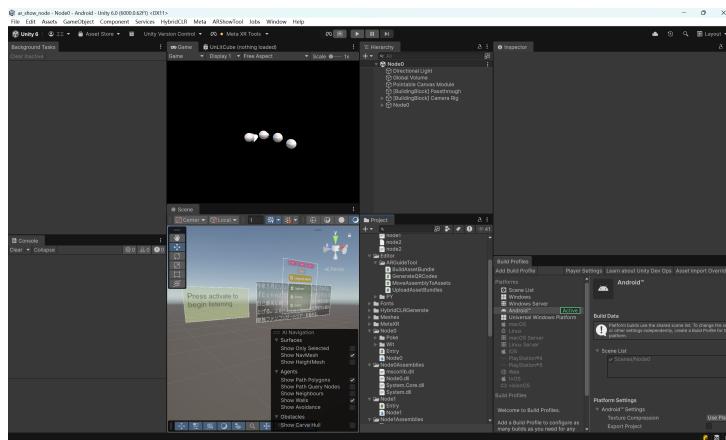


図 4.10 ARShowNodeGlobal [5].

HybridCLR

本システムにおける技術的な核心は、HybridCLR の採用にある。通常の Unity 製アプリ（IL2CPP ビルド）では、ビルド後に C# スクリプトの挙動を変更することは不可能である。しかし、HybridCLR を導入することで、C# コードをコンパイルした DLL をアセットとして扱い、実行時にインターペリタモードでロードして実行することが可能となる（図 4.11）。ARShowNode では、キュレーターが記述したスクリプトを HybridCLR によって AOT (Ahead-Of-Time) メタデータとホットアップデート用 DLL に変換し、これらを AssetBundle に含めることで、アプリ本体の更新を伴わない展示ロジックの配信を実現している。

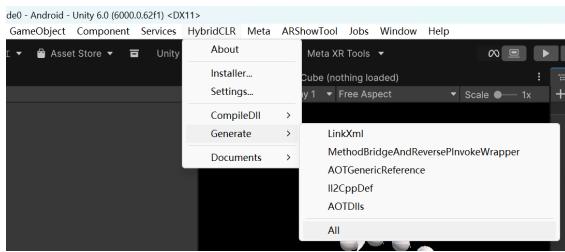


図 4.11 HybridCLRTool [5].

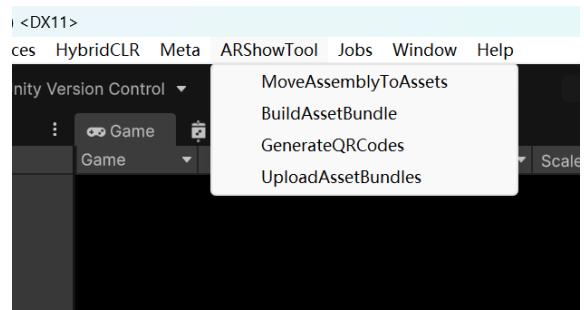


図 4.12 ARShowTool [5].

制作のための Unity ツール

本研究では、キュレーターの作業負荷を軽減するため、ARShowNode プロジェクト内に専用の Unity エディタ拡張ツールを実装した（図 4.12）。このツールはメニューバーからアクセス可能であり、以下の四つの機能を順次実行することで展示データの配信を行う。

1. **DLL の複製:** HybridCLR によって生成された AOT メタデータの DLL およびホットアップデートの DLL を、Unity プロジェクト内の Assets ディレクトリへコピーする。複数作品（Node）を同時制作する場合、各 Node に対応した DLL を複製する。
2. **AssetBundle のビルト:** 事前に設定されたラベル（Node0, Node1... 等の識別子）に基づき、各作品のリソースと DLL を含んだ AssetBundle を生成する。
3. **QR コード生成:** 各 AssetBundle の識別子（Node 名）情報を格納した QR コード画像を自動生成する。
4. **アップロード:** 生成された AssetBundle 群を、稼働中の静的ファイルサーバへ一括アップロードする。

Node

「Node」は、本システムにおける展示作品の単位であり、一つの AssetBundle に対応している。各 Node は、展示物の実体である Prefab、関連する設定ファイル、メディアアセット（画像、音響）、および制御用スクリプトのアセンブリの集合体である。本研究では、図 4.13 に示すように、同一 Unity プロジェクト内で複数の Node（Node0, Node1, Node2...）を並行して制作と管理可能な構造としている。

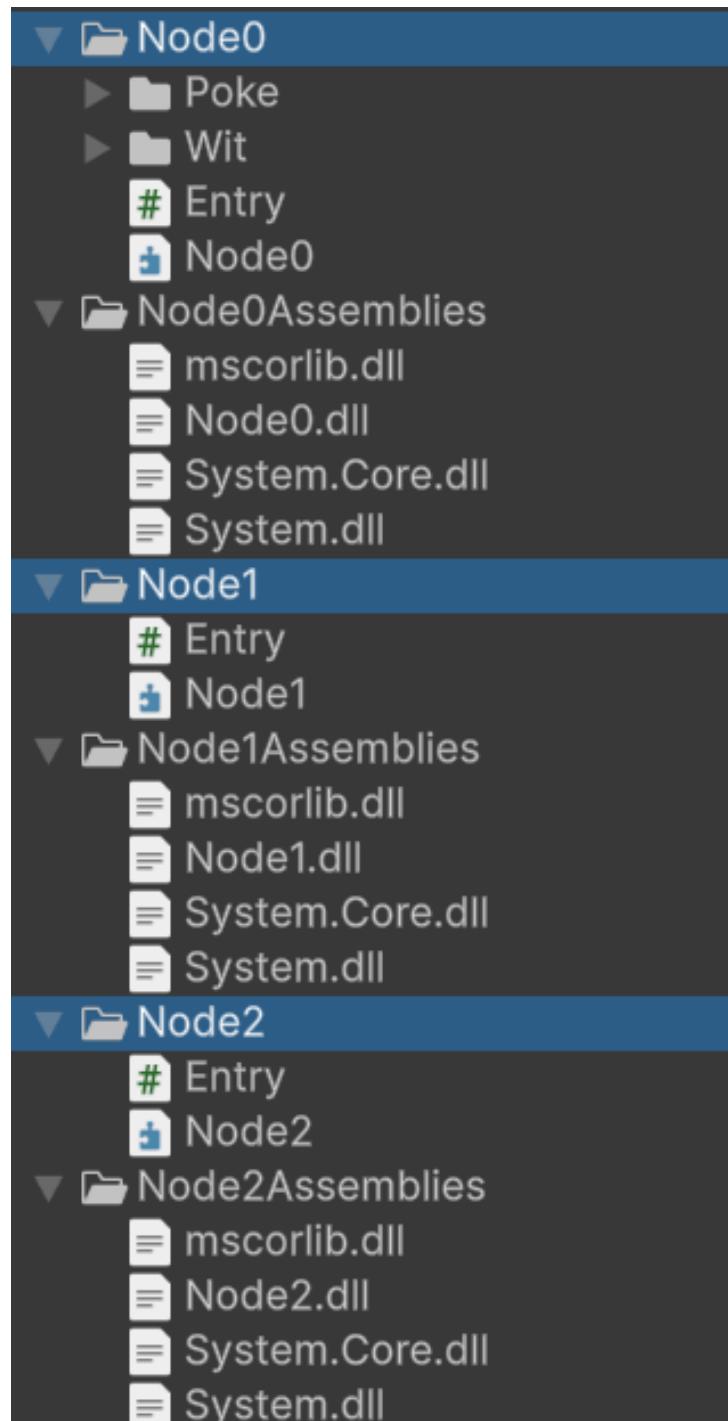


図 4.13 NodeGlobal [5].

エントリポイント (Entry.cs)

動的にロードされたスクリプトを、実行時 (Runtime) に正しくゲームオブジェクトにアタッチし機能させるため、本システムでは「Entry.cs」という規約に基づいたエントリポイントスクリプトを導入した（図 4.14）。

Entry.cs は各 Node の初期化ロジックを担い、AssetBundle のロード完了後、ARShow

アプリ側から明示的に呼び出されることで、Prefab のインスタンス化や依存コンポーネントのセットアップを実行する。実装コードの一部を図 4.15 に示す。

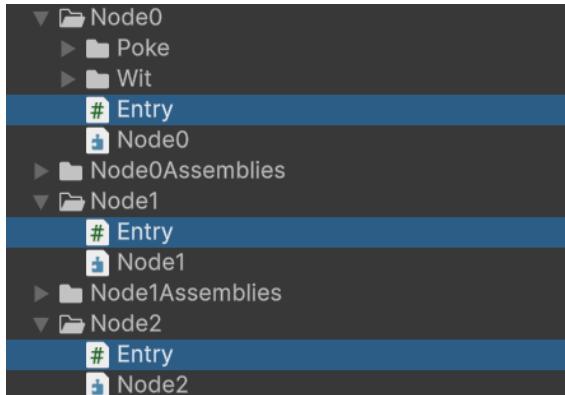


図 4.14 EntryGlobal [5].

```
/// <summary>
/// 热更新入口点方法
/// </summary>
/// <param name="assetBundle">从主项目传入的 AssetBundle 对象</param>
/// <returns>返回执行是否成功</returns>
0个引用
public static bool EntryPoint(object assetBundle, object loadedAssemblyDic,
{
    try
    {
        Debug.Log("[Entry.cs文件] Entry.EntryPoint 被调用");

        // 类型拆箱
        _ab = assetBundle as AssetBundle;
        _loadedAssemblyDic = loadedAssemblyDic as Dictionary<string, Assembly>;
        _instantiatedPrefabDic = instantiatedPrefabDic as Dictionary<string, GameObject>;
        _qrCodePose = pose as Pose?;
        // 参数验证
        if (_loadedAssemblyDic.Count == 0)
        {
            Debug.LogError("[Entry.cs文件] 传入的 loadedAssemblyDic 字典长度为0");
            return false;
        }
    }
}
```

図 4.15 EntryCode [5].

通常の Unity 開発では Inspector 上でスクリプトをアタッチするが、別プロジェクトでビルドされた AssetBundle 内のスクリプトを復元する場合、GUID の不整合等により参照が切れる問題がある。これを回避するため、本システムでは HybridCLR の推奨手法に基づき、実行時に「GameObject.AddComponent」メソッドを介してスクリプトを動的に付与する方式を採用した。

AssetBundle

Unity の AssetBundle 機能を利用し、Node ごとのリソースとコードをパッケージ化している。前述の通り、本システムでは AssetBundle 内に HybridCLR 用の DLL を含める点が特徴である（図 4.16）。これにより、3D モデルやテクスチャだけでなく、インタラクション等の仕組みも含めた完全な展示パッケージとして配信される。

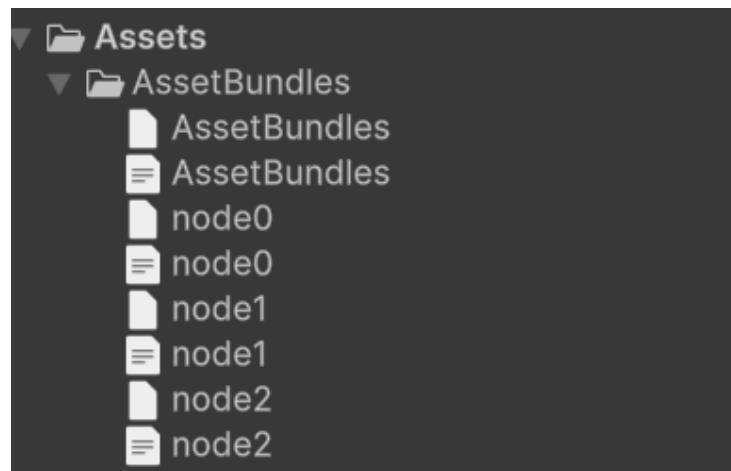


図 4.16 AssetBundleGlobal [5].

WitAI 配置

音声操作を実現するため、Meta 社が提供する自然言語処理サービス WitAI 配置を導入した（図 4.17）。Node 内で音声認識が必要な場合、ユーザーの音声入力は Meta XR Voice SDK を通じてテキスト化され、WitAI サーバへ送信される。サーバ上で事前に定義されたインテント（Intent）やエンティティ（Entity）の解析が行われ、その結果に基づいて Unity 側の関数がトリガーされる仕組みである。本研究では、中国語による音声コマンド制御の事例を実装した。

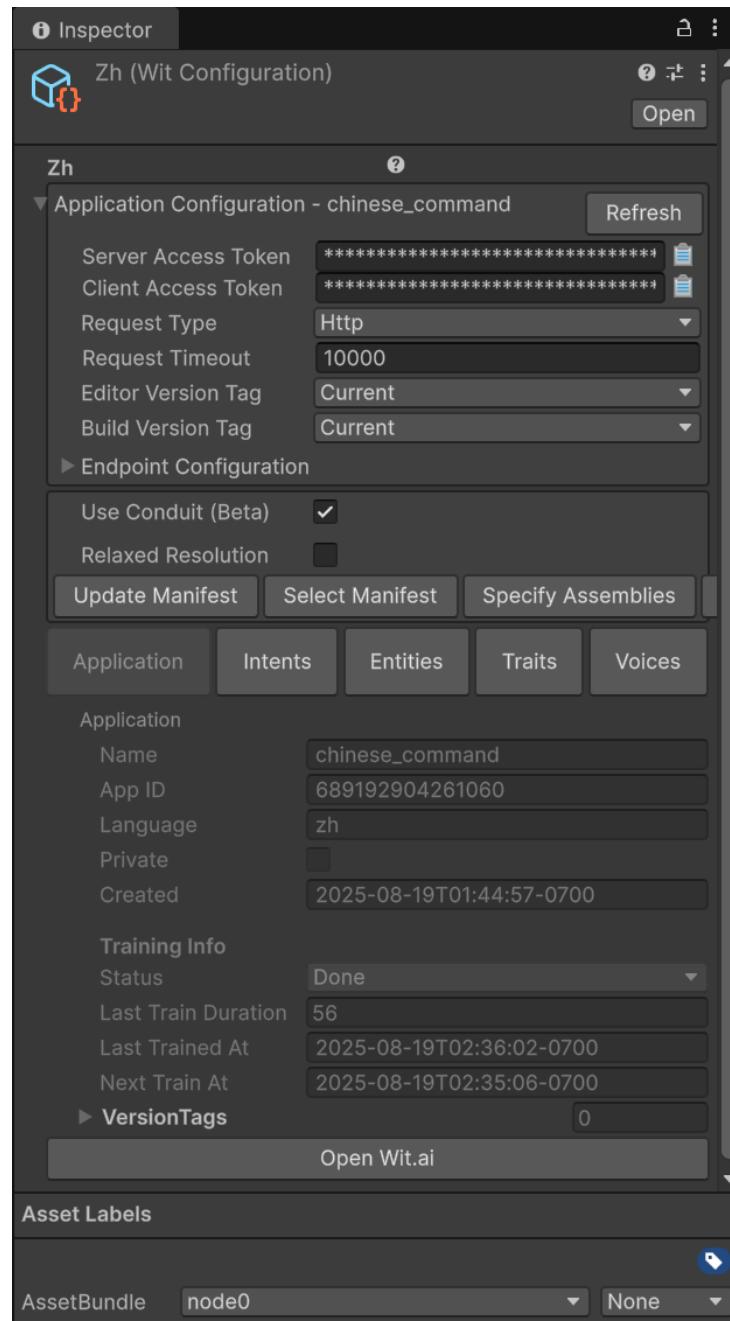


図 4.17 WitAI 配置 [5].

4.3.2 ARShow

ARShow は、Meta Quest 3 上で動作する鑑賞者用アプリケーションである（図 4.18）。本アプリは、QR コードのスキャン、AssetBundle のダウンロード、および動的コンテンツの再生環境を提供するコンテナとして機能する。

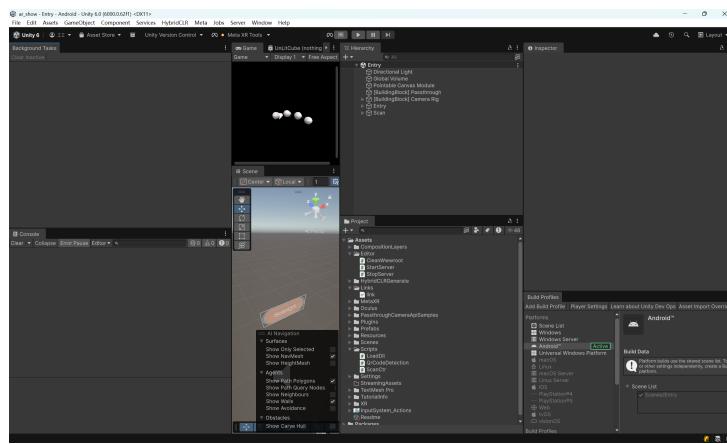


図 4.18 ARShowGlobal [5].

スキャンボタン

アプリ起動後、ユーザーは UI 上の「ScanQr」ボタンを押下することでスキャンモードへ移行する（図 4.19）。このモードでは、パススルーカメラの映像上に QR コード検出用のガイドが表示され、認識待機状態となる。

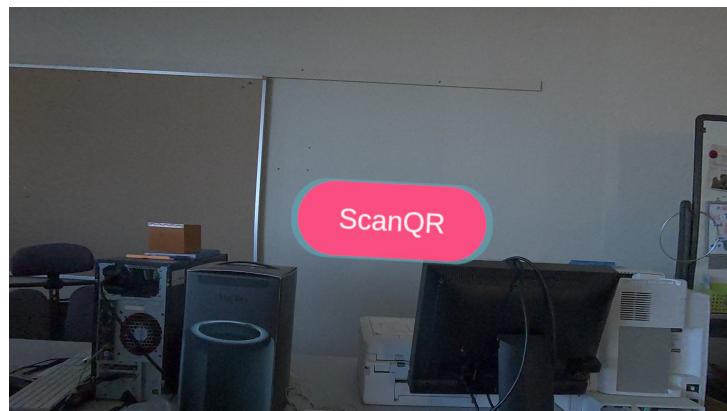


図 4.19 ScanQr ボタン [5].

QR Code 検出

QR コードの認識には ZXing.unity ライブラリを使用し、Quest 3 のカメラ映像からフレームごとの解析を行う。有効な QR コード（AssetBundle の識別子）が検出されると、スキャンモードを終了し、コンテンツのロード処理へ移行する。同一の QR コードを再ス

キャンした場合は、重複ロードを避けるためコンソールへ警告を表示する等の制御を行っている。

アセットロード

識別子に基づき、サーバから該当する AssetBundle をダウンロードする。ダウンロード中は、QR コードの物理位置にプログレスバー（進捗率）を空間表示し、ユーザーへのフィードバックを行う。初 AssetBundle のダウンロードプログレスが図 4.20 に示され、後続のものが図 4.21 に示される。

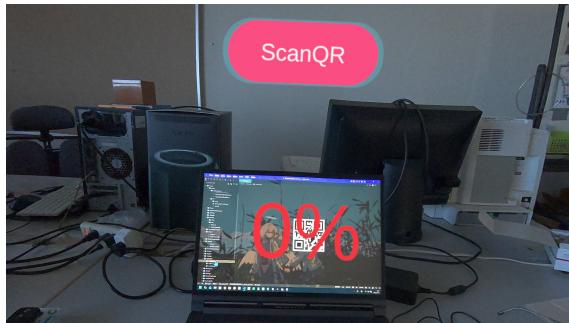


図 4.20 Node0Progress [5].



図 4.21 Node2Progress [5].

ダウンロード完了後、システムは以下の手順でアセットを展開する。

1. AssetBundle から HybridCLR 用の AOT メタデータ DLL をメモリ上に展開し、IL2CPP ランタイムに登録する。
2. ホットアップデート用 DLL をロードする。
3. ロードされたアセンブリ内から Entry.cs のエントリポイントメソッドをリフレクションを用いて呼び出す。

これにより、図 4.22 に示すように AR 展品がシーン内に初期化され、インタラクションが開始される。



図 4.22 Node0UI [5].

AOT ストリッピングの防止 (AOT Stripping Prevention)

Unity の IL2CPP ビルドでは、ビルド時に使用されていないコードや型情報はファイルサイズ削減のために削除（ストリッピング）される。しかし、動的にロードされる Node 側のスクリプトが、アプリ本体側で削除された型に依存している場合、実行時エラーが発生する。

これを防ぐため、本システムでは Link.xml ファイルを定義し（図 4.23）、Node で使用する可能性のあるアセンブリや型を明示的に記述することで、ビルド時のストリッピングを回避している。これにより、ARShowNode で開発された任意のスクリプトが、ARShow アプリ上で正しく動作することを保証している。



```
Assets > Links > link.xml
<linker>
  <assembly fullname="UnityEngine.VideoModule">
    <type fullname="UnityEngine.Video.VideoPlayer" preserve="all"/>
    <type fullname="UnityEngine.Video.VideoClip" preserve="all"/>
  </assembly>
  <assembly fullname="Meta.WitAi" preserve="all"/>
  <assembly fullname="Meta.Voice" preserve="all"/>
  <assembly fullname="Oculus.Voice" preserve="all"/>
  <assembly fullname="VoiceSDK.Runtime" preserve="all"/>
  <assembly fullname="UnityEngine.AssetBundleModule" preserve="all"/>
</linker>
```

図 4.23 Linkxml 配置 [5].

4.3.3 サーバ

本システムにおけるサーバは、動的なプログラム処理（アプリケーションロジック）を持たず、クライアントからのリクエストに応じてファイル（AssetBundle）を配信することに特化した静的ファイルサーバである。実装には ASP.NET Core を採用しており、未来に機能（ユーザ情報登録等）を拡張するために柔軟性を保持してある。ホストが開発 PC 機上で動作する。これにより、サーバサイドとクライアント（Unity）間で C# によるプログラム言語を統一させ、開発効率の向上を図っている。

「ARShow」プロジェクトのメニューバーには、図 4.24 に示すように、このサーバの「起動」「停止」「ディレクトリ清掃」を制御する機能も統合されており、展示運用時のサーバ管理を容易にしている。なお、開発環境の LAN（構内通信網）においては、ARShow アプリにサーバから AssetBundle をダウンロードする用の IP アドレスを開発 PC 機の実際アドレスと一致させる必要がある。

4.4 アーキテクチャ

提案システムはクライアント・サーバモデルを採用し、以下の三つの主要コンポーネントから構成される（図 4.25）。

- **ARShowNode:** AR 展示物の作成とアップロードを行うキュレーター側プログラム
- **ARShow:** 閲覧者用クライアントアプリケーション

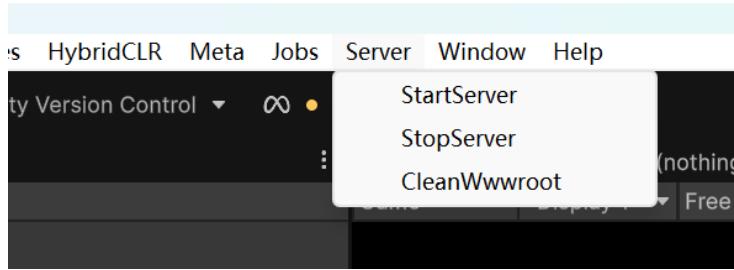


図 4.24 ServerTool [5].

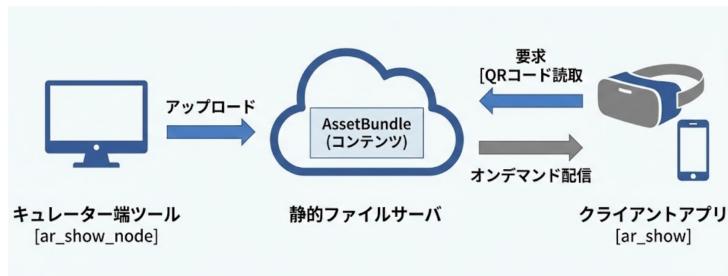


図 4.25 提案システムの全体構成図 [5].

- **Static File Server:** コンテンツを配信する静的ファイルサーバ

この 3 つの部分が有機的に連携することにより、キュレーターは自身のコンテンツを容易に配信かつ更新でき、閲覧者は最新の展示物をシームレスに体験できる環境が実現される。

展示作品の実体である AssetBundle は、アプリケーション内部には保持されず、外部サーバに配置される。クライアントアプリは QR コードの読み取りをトリガーとして、必要なデータのみをオンデマンドで取得し展開する。このアーキテクチャにより、鑑賞者のデバイストレージを圧迫することなく、作品数を無制限に拡張することが可能となる。また、展示内容の更新はサーバ側のアーカイブファイル差し替えのみで完結するため、前述した「更新プロセスの困難さ」および「コミュニケーションコスト」の課題を解決する基盤となる。

4.5 UI とインタラクション

本節では、本システムにおいて実装された三つ代表的な展示作品（Node）を例に挙げ、鑑賞者が体験するユーザーインターフェース（UI）およびインタラクションの詳細について述べる。各 Node はそれぞれ異なるメディア形式（複合 UI、映像、3D モデル）を扱っており、システムの汎用性を示している。

4.5.1 Node0: 複合的な AR 展示インターフェース

Node0 は、本システムの中で最も機能的に複雑な展示例であり、文化財の 3D モデル表示と解説テキスト、および音声操作を組み合わせた複合的な AR コンテンツである。実際の UI 表示を図 4.26 に示す。

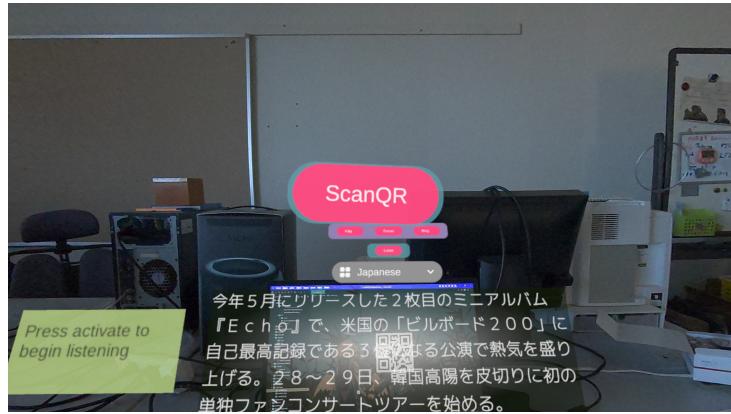


図 4.26 Node0UI [5].

インターフェース構成

本 Node の UI は、Meta Horizon OS UI Set をベースに構築されており、Meta Quest の標準的なシステム UI と親和性の高いデザインを採用している。画面構成は主に「音声のテキスト化フィードバック」と「操作用テキストキャンバス」の二つのモジュールから成る。テキストキャンバスは、機能に応じて以下の三つのセクションに区分されている。

- ・スクリーンリーダー制御: 図 4.27 に示すように、解説音声の「再生 (Play)」「一時停止 (Pause)」「停止 (Stop)」を行うボタン群が配置されている。
- ・多国言語制御: 文化財の解説テキストを表示するエリアおよび言語選択ドロップダウンメニューである。言語切り替えにより、テキストと読み上げ音声が即座に変更される。
- ・音声コマンド制御: 「Listen」ボタンと認識結果を表示するテキストボックスから構成される（図 4.28）。

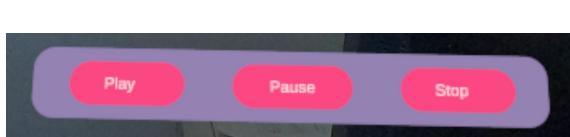


図 4.27 Node0UIReader [5].

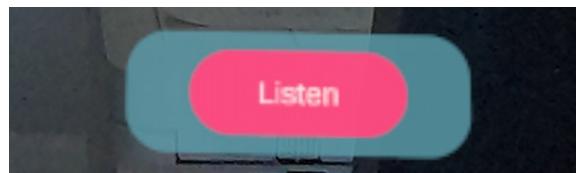


図 4.28 Node0UIListen [5].

インタラクション

鑑賞者は、ハンドトラッキング機能を用いた以下のように直感的な操作が可能である。

- **タッチ操作 (Poke)**：仮想ボタンを指先で直接押すことで、再生制御やモード切替を行う（図 4.29）。
- **遠隔操作 (Ray)**：手から発せられるレイ（光線）を用いて、離れた位置にある UI 要素を選択と操作する（図 4.30）。
- **把持操作 (Grab)**：UI パネルや 3D モデルを「掴む」ジェスチャにより、鑑賞しやすい位置や角度へ自由に移動させることができる（図 4.31）。

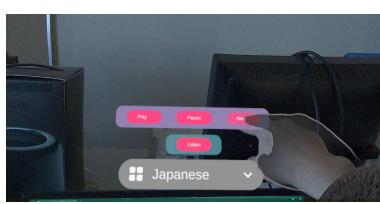


図 4.29 Node0UIPoke [5].

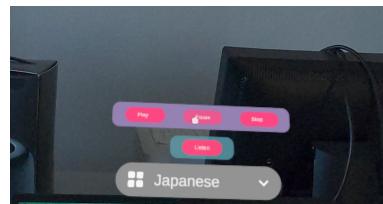


図 4.30 Node0UIRay [5].

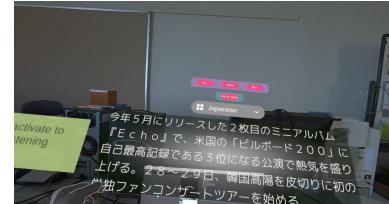


図 4.31 Node0UIGrab [5].

これらのインタラクションを実現するため、技術的には PointableCanvasModule システムを利用している（図 4.32）。通常、このイベントシステムは Unity シーン内に常駐する必要があるが、本システムでは動的にロードされる Node 側（ARShowNode）で定義されたイベント設定を、実行時に本体アプリ（ARShow）のシーンへ正しく引き継ぐ仕組みを実装することで、スムーズな操作性を確保している。

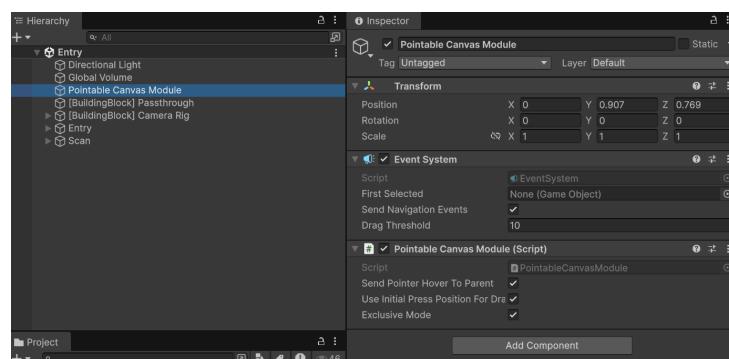


図 4.32 PointableCanvasModule [5].

音声コマンド制御

「Listen」ボタンを押下すると音声認識モードへ移行し、Wit.ai を介したボイスコマンドによる操作が可能となる。図 4.33 と図 4.34 に示すように、システムは待機状態（Ignore）から聞き取り状態（Listen）へと遷移する。

例えば、所定のキーワードを発話することで、ボタン操作なしに解説の再生や言語変更を行うことができる。認識された発話内容は左側 UI モジュールにテキストとしてフィー

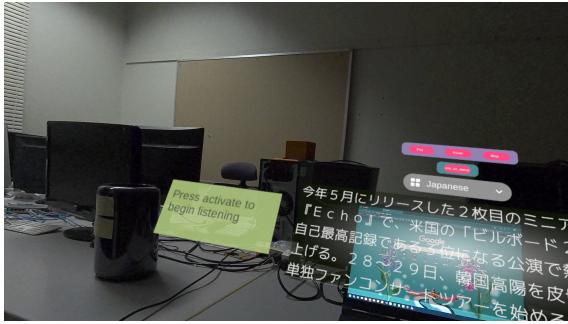


図 4.33 Node0VoiceIgnoreStatus [5].



図 4.34 Node0VoiceListenStatus [5].

ドバックされ、確実な入力を支援する。

図 4.35 と図 4.36 に中国語および英語の言語に切り替えた結果を示す。



図 4.35 Node0VoiceNviCh [5].

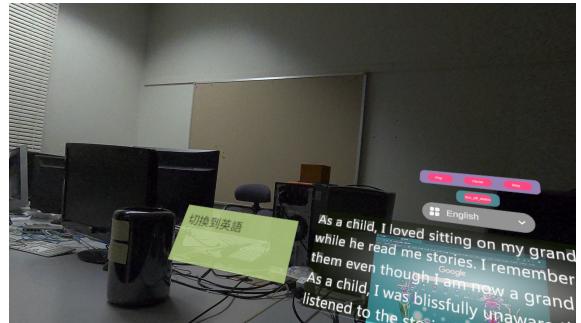


図 4.36 Node0VoiceNviEn [5].

図 4.37 と図 4.38 に音声コマンドによる再生開始と停止の挙動を示す。

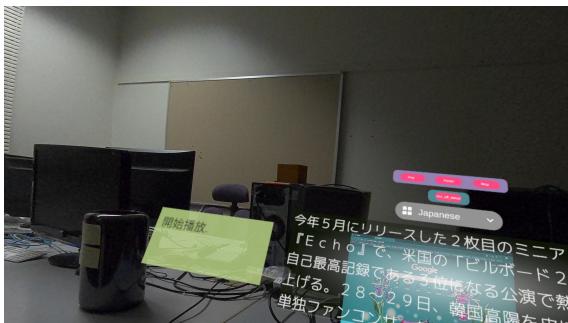


図 4.37 Node0VoicePlayStart [5].



図 4.38 Node0VoicePlayStop [5].

4.5.2 Node1: 映像コンテンツの空間配置

Node1 は、映像メディアを AR 空間内に配置する展示例である（図 4.39）。Unity のプリミティブ形状である Quad（板ポリゴン）に Video Player コンポーネントを付加し、独自の録画映像と音声を再生する構成となっている。本 Node にも Grab インタラクションが付与されており、鑑賞者は空中に浮かぶスクリーンを手に取り、壁面に配置したり、目

の前に引き寄せて細部を確認したりといった、物理的なスクリーンと同様の取り回しが可能である。これは、動画解説パネルとしての利用を想定した実装である。



図 4.39 Node1UI [5].

4.5.3 Node2: 3D モデル（文化財）の展示

Node2 は、静的な 3D オブジェクトの展示に特化した最小構成の例である（図 4.40）。ここでは、Sketchfab より取得した青銅器の 3D モデル（glTF 形式）を Prefab 化し、AssetBundle として配信している。Node1 同様、Grab インタラクションが設定されており、鑑賞者は貴重な文化財モデルを仮想的に手に取り、あらゆる角度から詳細に観察することができる。この Node は、複雑なスクリプトを含まない純粋なアセットデータも、本システムを通じて問題なく配信や操作可能であることを実証している。

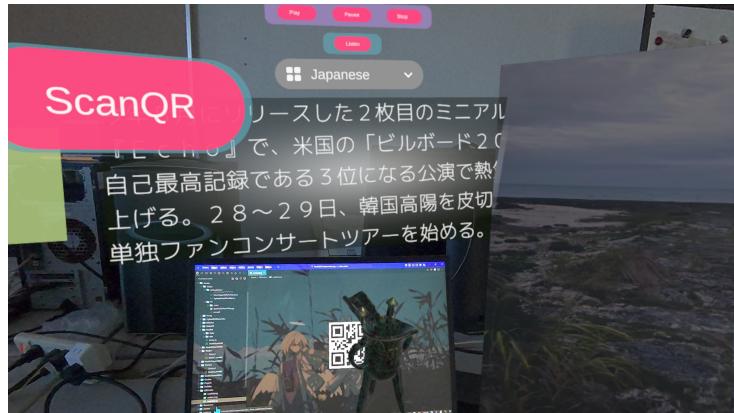


図 4.40 Node2UI [5].

4.6 ワークフロー

本節では、本システムを用いた AR 展示の制作から鑑賞に至るまでの具体的なワークフローについて、制作サイド（キュレーター）と鑑賞サイド（閲覧者）の双方の視点から述べる。

4.6.1 制作サイド（キュレーター）

キュレーター側の作業は、Unity プロジェクト「ARShowNode」を用いて行われる。制作から公開に至る全体のワークフローを図 4.41 に示す。

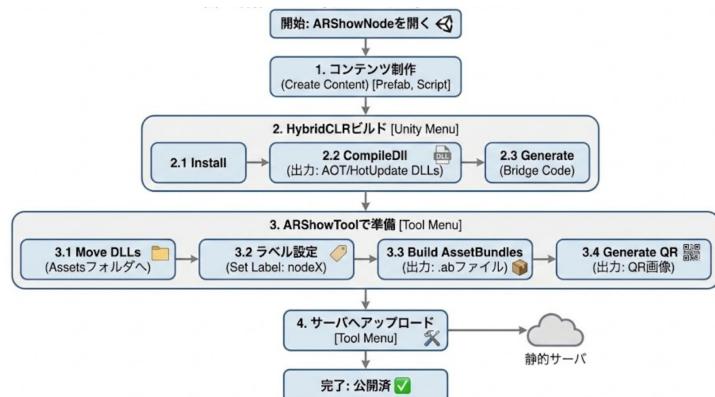


図 4.41 制作者ワークフロー [5].

ARShowNode プロジェクトの配置

まず、キュレーターは提供される「ARShowNode」プロジェクトを開発環境に展開する。このプロジェクトには、Meta XR All-in-One SDK、HybridCLR、および本研究で開発した専用ツールキット等の依存ライブラリが事前設定されている。キュレーターは、

Unity エディタ上でコンパイルエラーがない状態を確認し、自身のコンテンツ（Prefab やスクリプト）の制作を開始する。

Assembly のビルド

作品のロジック（C# スクリプト）をホットアップデート可能な形式に変換するため、HybridCLR のビルド機能を実行する。まず、HybridCLR メニューから Install を実行し、環境を初期化する。次に CompileDll コマンドを実行し、ターゲットプラットフォーム（Android）向けの DLL（AOT メタデータおよびホットアップデート用アセンブリ）を出力する。そして Generate コマンドを実行し、ブリッジコード等を生成する。

AssetBundle の事前準備

DLL のビルド完了後、上述の本研究が提供した Unity ツール（ARShowTool）を用いて、以下の手順で配信準備を行う。

1. **DLL の配置:** ツールメニューの「Move DLLs」を実行し、生成された DLL ファイルを Unity プロジェクトの Assets フォルダへ複製する。
2. **バンドル設定:** 各作品のリソース（Prefab, DLL 等）に対し、一意の AssetBundle ラベル（例: node0）を付与する。
3. **ビルド実行:** ツールメニューの「Build AssetBundles」を実行する。これにより、ラベル付けされたリソースが一つの AssetBundle ファイルとしてパッケージ化される。
4. **QR コード生成:** ツールメニューの「Generate QR」を実行し、各 AssetBundle に対応する QR コード画像を生成する。

サーバへのアップロード

最後に、ツールメニューの「Upload to Server」を実行する。これにより、生成された全ての AssetBundle ファイルが、LAN（構内通信網）内で稼働している静的ファイルサーバの公開ディレクトリへ自動的に転送される。以上で、展示コンテンツの公開作業は完了である。

4.6.2 鑑賞サイド（鑑賞者）

鑑賞者は、HMD（Meta Quest 3）を装着し、図 4.42 に示す手順で展示を体験する。

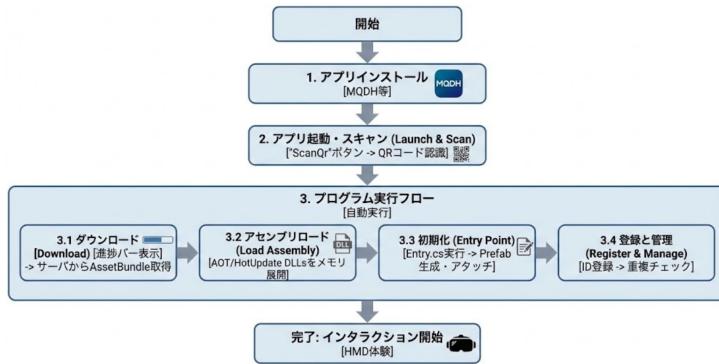


図 4.42 閲覧者ワークフロー [5].

ARShow AR APP のインストール

事前に、Meta Quest Developer Hub (MQDH) 等を経由して、閲覧用アプリ「ARShow」をデバイスにインストールする。

スキャンモードによる開始

アプリを起動すると、空間上に「ScanQr」ボタンが表示される。これをクリックすると、パススルーカメラを用いたQRコードスキャンモードに移行する。鑑賞者が展示会場に掲示されたQRコードに視線を向けると、システムは自動的にコードを認識して解析する。

プログラムの実行フロー

QRコードの認識後、システムは以下のフローを自動的に実行する。

- 1. ダウンロード:** 解析された ID に基づき、サーバから対応する AssetBundle をダウンロードする。進捗状況は空間上のプログレスバーで可視化される。
- 2. アセンブリロード:** ダウンロード完了後、AssetBundle 内の AOT メタデータ DLL とホットアップデート DLL をメモリに展開する。
- 3. 初期化 (Entry Point) :** ロードされたアセンブリ内の Entry.cs を特定し、その初期化メソッドを実行する。この段階で、展示作品の Prefab がシーン内に生成 (Instantiate) され、必要なコンポーネントが動的にアタッチされる。
- 4. 登録と管理:** 生成された作品は ID と共に内部辞書に登録される。スキャンモードは終了し、鑑賞者は作品とのインタラクションが可能となる。

なお、既にロード済みの QR コードを再スキャンした場合は、重複ロードを防ぐためコンソールに警告が表示され、およびスキャンモードが解除される仕様となっている。

第 5 章

評価方針

本章では、前章で提案した「ARShowNode」および「ARShow」から成る AR 展示プラットフォームの実用性と有効性を検証するための実験計画について述べる。本実験では、実際の展示運用を模した一連のワークフローを被験者に実施させ、システムユーザビリティの観点から定量的な評価を行うとともに、自由記述による定性的なフィードバックを収集し、システムの有用性および改善点を明らかにすることを目的とする。

5.1 実験仮説

本研究は、HMD を用いた AR 展示において、制作者（キュレーター）のコンテンツ更新負荷を軽減し、かつ閲覧者（鑑賞者）に対して直感的で没入感のある鑑賞体験を提供することを目的としている。この目的に基づき、本実験では以下の三つの仮説を立証することを試みる。

5.1.1 仮説 1: 制作プロセスの効率化

提案システムが提供する「ARShowTool」およびホットアップデート技術を用いることで、制作者はアプリケーション本体の再ビルドを行うことなく、短時間かつ低成本で複雑なロジックを含む AR コンテンツの配信が可能である。これにより、従来のアプリ配布型展示と比較して運用コストが著しく低減される。

5.1.2 仮説 2: 鑑賞体験のアクセシビリティ向上

QR コードを起点とした物理空間と情報空間のシームレスな接続に加え、ハンドトラッキングによる身体的な操作（把持や指差し）および音声コマンドによる自然言語入力を統合することで、HMD 特有のコントローラ操作を排除した直感的な操作体系を実現する。このマルチモーダルなインタラクション設計により、鑑賞者は複雑なメニュー操作や事前の導入手順を学習することなく、自然な挙動で AR 体験を開始および操作できる高いアク

セシビリティを有する。

5.1.3 仮説 3: システム全体の統合的ユーザビリティ

制作者と閲覧者の双方が、それぞれの役割においてシステムを円滑に操作可能であり、Unity ツールによるコンテンツ生成からサーバ配信、そしてクライアントアプリによる鑑賞に至る一連のプロセスが、技術的な破綻や遅延なく統合的に機能する。これにより、本システムが単なるプロトタイプに留まらず、実用的な展示プラットフォームとしての実現可能性を有していることが示される。

5.2 評価手法

5.2.1 定量的評価 (SUS)

本実験における定量的評価指標として、System Usability Scale (以下、SUS) を採用した。SUS は、Brooke (1996) によって考案されたユーザビリティ評価のための標準的なアンケート手法であり、システムの有効性、効率性、満足度を包括的に測定することが可能である。

SUS を採用した理由は以下の 2 点である。

- ・ 第一: 技術的なシステムから一般的な製品まで幅広い対象に適用可能であり、学術的にも信頼性と妥当性が広く認められている点である。
- ・ 第二: 10 項目の質問で構成されており、被験者への負担を最小限に抑えつつ、比較可能なスコアを算出できる点である。

質問項目は 5 段階のリッカート尺度 (1: 全くそう思わない ~ 5: 強くそう思う) で回答を求める。なお、質問文中の「システム」という用語は、被験者の役割に応じて「AR 制作ツール (ARShowNode)」または「AR 閲覧アプリ (ARShow)」と読み替えるよう教示を行う。本実験で使用する SUS の質問項目を表 5.1 に示す。

5.2.2 定性的評価

SUS による定量評価に加え、自由記述形式のアンケートを実施する。ここでは、操作中に感じた具体的な困難点、システムの挙動に関する違和感、および将来的な機能要望について記述を求め、SUS 数値には現れないユーザビリティの課題を抽出する。

5.3 被験者の構成

本実験の被験者は、Google フォームを用いた公募により選定する。実験の性質上、システムの「制作側」と「閲覧側」双方の視点が必要となるため、計 4 名の被験者を採用し、それぞれ以下の役割を割り当てる。

表 5.1 本実験で使用する System Usability Scale (SUS) の質問項目

No.	質問内容
1	このシステムを頻繁に使用したいと思うか。
2	このシステムは必要以上に複雑だと感じるか。
3	このシステムは使いやすいと感じるか。
4	このシステムを使用するために、技術的なサポートが必要だと感じるか。
5	このシステムの様々な機能はよく統合されていると感じるか。
6	このシステムには一貫性がない（矛盾が多い）と感じるか。
7	多くの人々がこのシステムの使い方をすぐに習得できると思うか。
8	このシステムは操作が非常に煩雑（面倒）だと感じるか。
9	このシステムを使っていて安心感（自信）を持てるか。
10	このシステムを使い始める前に、多くのことを学ぶ必要があると感じるか。

5.3.1 制作者 (Creator)

- 人数: 2 名
- 選定条件: Unity エンジンの基本的な操作（エディタ操作、C# スクリプトの理解）に習熟していること。
- 選定理由: 提案システム「ARShowNode」は Unity 開発者を対象としたツールキットであるため、一定の開発スキルを有するユーザーによる専門的な評価が不可欠であるため。

5.3.2 閲覧者 (Viewer)

- 人数: 2 名
- 選定条件: 特段の技術的背景は問わないが、HMD デバイスの使用に抵抗がないこと。
- 選定理由: 一般の美術館や展示会への来場者を想定し、専門知識を持たないユーザーでも直感的に操作可能かを検証するため。

このように役割を分担することで、実際の展示会における「キュレーター（作品提供者）」と「オーディエンス（鑑賞者）」の関係性を模倣し、システム全体のエコシステムとしての妥当性を検証することを意図する。

5.4 操作説明

実験に先立ち、各被験者に対して以下の通り操作説明（教示）を行う。

5.4.1 制作者への操作説明

制作者役には、Unity プロジェクト「ARShowNode」および本研究で実装した「ARShowTool」の使用方法を説明する。以下の六つの手順を順次実行することで、コンテンツの配信が完了することを教授する。

1. **DLL のコンパイル (CompileDll)** : HybridCLR メニューから「CompileDll」を実行し、ターゲットプラットフォーム (Android) に対応したホットアップデート用 DLL を出力する。
2. **コード生成 (GenerateAll)** : 続いて HybridCLR メニューから「GenerateAll」を実行し、AOT メタデータおよびブリッジコードを生成する。これにより、アプリ本体を更新することなくロジック配信を可能にする。
3. **アセンブリの配置 (MoveDLLs)** : ARShowTool メニューの「MoveDLLs」を実行し、生成された DLL ファイルを Unity プロジェクト内の Assets ディレクトリへ自動的に複製する。
4. **アセットバンドルの構築 (BuildAssetBundles)** : ARShowTool メニューの「BuildAssetBundles」を実行する。事前に設定されたラベル（例: node0, node1）に基づき、リソースと DLL を含んだ AssetBundle ファイルを生成する。
5. **サーバへの配信 (UploadToServer)** : 最後に ARShowTool メニューの「UploadToServer」を実行する。生成された AssetBundle 群が、稼働中の静的ファイルサーバへ一括転送され、配信状態となる。

5.4.2 閲覧者への操作説明

閲覧者役には、Meta Quest 3 にインストールされた「ARShow」アプリの操作方法を説明する。アプリはパススルー機能を用いた AR モードで動作し、以下の手順で鑑賞を行う旨を伝える。

1. **スキャンモードの開始:** アプリ起動後、空間上に表示される「ScanQR」ボタンを押下 (Poke) し、QR コード認識待機モードへ移行する。
2. **QR コードの検出:** パススルー映像越しに、制作者から提示された QR コードに視線を向ける。認識されると自動的にロード処理が開始される。
3. **ダウンロードと展開:** サーバからの AssetBundle ダウンロード、HybridCLR による DLL ロード、および Entry ポイントの実行が自動的に行われ、目の前に展品が出現する。
4. **インタラクション:** 出現した展品に対し、ハンドトラッキングを用いた操作 (Grab、Ray、Poke、音声コマンド) をを行い、鑑賞する。

5.5 実験手順

本実験では、実験における外的要因を排除し、純粋に提案ツールの有用性と AR 展品の鑑賞体験を評価するため、制作者による「新規 Unity プロジェクトの作成」工程は省略し、ホスト（実験実施者）が以下の環境準備を行うものとする。

- **事前準備:** 開発用 PC に「ARShowNode」プロジェクトをセットアップし、静的ファイルサーバを起動する。
- **環境の初期化:** 各実験セットの開始前に、前回のデータによる干渉を防ぐため、生成物（AssetBundles、QR コード）およびサーバ上のファイルを完全に削除し、初期状態にリセットする。

ハードウェアリソースの制約（PC 1 台、HMD 1 台）を考慮し、被験者を二つのグループに分け、図 5.1 に示す流れで順次実験を行う。

5.5.1 グループ A（制作者 1、閲覧者 1 2）

1. **制作フェーズ:** 制作者 1 は、ARShowTool を用いて二つの異なる展品「Node0（解説付きキャンバス）」および「Node1（映像展示）」のビルドからアップロードまでを行う。生成された二つの QR コードを画面表示により閲覧者に提供する。
2. **閲覧フェーズ:** 閲覧者 1 が Quest 3 を装着し、Node0、Node1 の順に QR コードをスキャンして体験を行う。体験終了後、HMD を閲覧者 2 に渡し、同様に体験を行う。

5.5.2 グループ B（制作者 2、閲覧者 1 2）

1. **制作フェーズ:** 制作者 2 は、同様の手順で「Node0」および「Node2（静的 3D モデル）」のビルドとアップロードを行う。これにより、異なるデータタイプの処理に対する安定性を検証する。
2. **閲覧フェーズ:** グループ A と同様に、閲覧者 1、閲覧者 2 の順で Quest 3 を装着し、Node0 および Node2 の体験を行う。

全ての体験が終了した後、4 名の被験者は Google フォームにて SUS アンケートおよび自由記述によるフィードバックへの回答を行う。

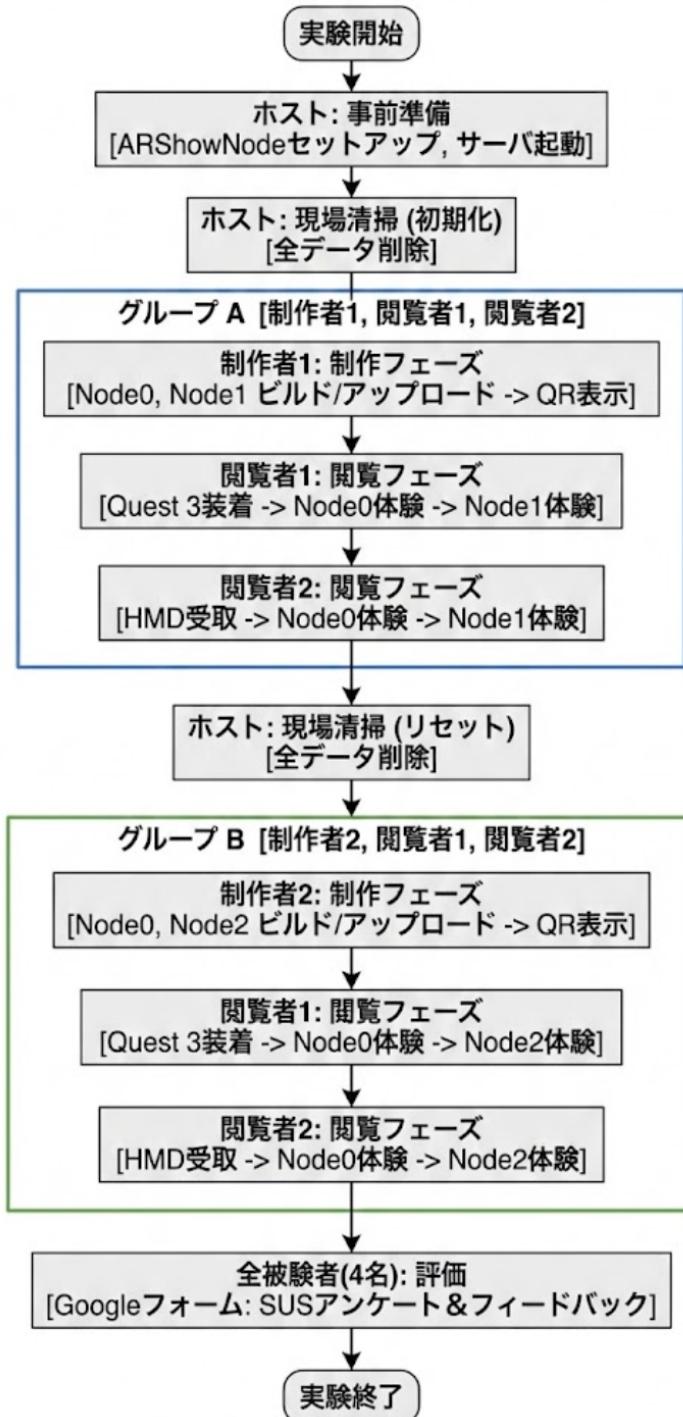


図 5.1 本実験の実施フロー [5].

第6章

まとめ

6.1 総括

本研究では、独立系キュレーターによる AR 空間芸術展示の普及に伴い顕在化した、展示運用におけるコミュニケーションコストの増大と、コンテンツ更新プロセスの硬直性という課題に着目した。

従来のアプリケーション配布型の展示手法では、鑑賞者に対する導入支援の負担が大きく、また展示ロジックの些細な修正であってもアプリケーション全体の再ビルトと再配布を要するため、柔軟な展示運用が困難であった。そこで本研究では、HMD と QR コードを活用し、制作者（キュレーター）と閲覧者（鑑賞者）をシームレスに接続する中間プラットフォームを提案および開発した。

本研究の主な成果は以下の通りである。

- 第一: Unity の AssetBundle 機能と HybridCLR フレームワークを統合することで、従来困難とされていた C# スクリプトのホットアップデートを実現するアーキテクチャを構築した。これにより、制作者はアプリケーション本体を更新することなく、3D モデルのみならずインタラクションロジックを含めた展示コンテンツを動的に配信かつ更新することが可能となった。また、制作者向けに実装した Unity 拡張ツール「ARShowTool」により、DLL のビルトから AssetBundle の生成、サーバへのアップロードまでの一連の工程を自動化し、制作プロセスを大幅に効率化した。
- 第二: Meta Quest 3 のパススルー機能とハンドトラッキング、および音声認識技術 (Wit.ai) を組み合わせた、直感的かつ没入感の高い鑑賞環境を実現した。鑑賞者は会場の QR コードをスキャンするだけで、必要なコンテンツをオンデマンドで取得し展開できるため、複雑なアプリ操作や事前設定を不要とした。また、開発したプロトタイプにおいて、複合 UI、映像、3D モデルという異なるメディア形式の展示が可能であることを示し、システムの汎用性を確認した。
- 第三: 制作者と閲覧者の役割を分担した被験者実験を通じて、SUS による定量的評

価および自由記述による定性的評価を行う方針を定めた。本システムは、LAN（構内通信網）環境下における動作検証において、意図通りのコンテンツ配信と動的ロードが機能することを確認できており、実用的な展示プラットフォームとしての基盤確立を達成したといえる。

6.2 今後の課題

本研究で提案したシステムの実用化に向けた今後の課題として、以下の点が挙げられる。

6.2.1 評価実験の実施と検証

本論文執筆時点において、前章で設計された評価実験はまだ未実施である。そのため、実際の被験者を対象とした実験を行い、以下の仮説を検証する必要がある。具体的には、制作者にとっての「制作と更新プロセスの効率化」、鑑賞者にとっての「アクセシビリティの向上」、そしてシステム全体の「統合的ユーザビリティ」である。定量的と定性的なデータを収集し分析することが急務である。

6.2.2 コンテンツの安全性審査

本システムは、制作者が作成した任意のスクリプトやアセットを実行時にロードする仕組みであるため、配信されるコンテンツの品質管理が重要な課題となる。現在、アップロードされた AssetBundle に対する審査機構は実装されていない。しかし、一般公開を想定した場合、公序良俗に反するコンテンツや、激しい光の点滅など鑑賞者の健康を害する恐れのある演出（光過敏性発作の誘発リスク等）が含まれる可能性がある。したがって、サーバへのアップロード時に自動でメタデータを解析するフィルタリング機能や、運営者による事前承認プロセスの導入など、コンテンツの安全性と健全性を担保する仕組みの検討が必要である。

6.2.3 セキュリティの強化

HybridCLR を用いたホットアップデートは強力な機能である反面、セキュリティリスクを伴う。悪意のある制作者が、デバイス内の個人情報にアクセスするスクリプトや、システムをクラッシュさせるコードを AssetBundle に混入させるリスクが考えられる。現状のシステムは信頼できる制作者（Closed 環境）を前提としているが、プラットフォームを開放するにあたっては、実行可能な C# API の制限（サンドボックス化）や、DLL に対する電子署名による改ざん検知機能の実装が不可欠である。

6.2.4 クラウド環境への移行とスケーラビリティ

現在の実装は LAN（構内通信網）内の運用を前提とした静的ファイルサーバ構成であるが、大規模な展示会や遠隔地からのアクセスに対応するためには、クラウドインフラ

への移行が必要である。AWS や Cloudflare 等の CDN（コンテンツデリバリネットワーク）を活用し、地理的に分散したアクセスに対しても高速に AssetBundle を配信できるアーキテクチャへの拡張が求められる。また、同時に多数の鑑賞者が QR コードをスキャンした際のサーバ負荷分散や、ネットワーク帯域の最適化についても検討の余地がある。

謝辞

本研究の遂行、ならびに本論文の執筆にあたり、ご多忙にも関わらず終始懇切なるご指導とご鞭撻を賜りました、指導教員である赤嶺有平先生に深く感謝の意を表します。先生の熱意あるご指導は、私の研究に対する姿勢の礎となりました。

また、本研究の方向性や内容に対し、多角的な視点から貴重なご助言とご指導を賜りました、副指導教員の當間愛晃先生、國田樹先生に深く感謝いたします。

二年間の研究生活において、有意義な議論と温かい励ましをくれた知能情報コース赤嶺研究室の皆様に感謝いたします。特に、私の研究活動に対して多大なる尽力と助力をいただいた HYUN JUWON 先輩には、心より御礼申し上げます。また、共に研究室を盛り上げ、私の研究を支えてくれた修士課程の後輩たち、学部4年生の皆様にも深く感謝いたします。

最後に、修士課程の期間中、私が最も困難な時期にあっても変わらぬ愛情で精神的に支え続け、また物心両面にわたる多大なる支援をしてくれた家族に、最大の感謝を捧げます。家族の献身的な支えなくして、本研究を完遂することは叶いませんでした。ここに心からの感謝の意を表します。

2026年3月
閻永祥

参考文献

- [1] Ohlei, A., Schumacher, T., & Herczeg, M. (2020). An Augmented Reality Tour Creator for Museums with Dynamic Asset Collections. In Augmented Reality, Virtual Reality, and Computer Graphics (LNCS 12243, pp. 15-31). Springer.
- [2] Duanmu, Q., Dai, T., Cai, Y., & Herman, J. (2023). AR MUSE: Designing and Implementing a Solution for Accessible Augmented Reality Exhibition. Worcester Polytechnic Institute.
- [3] Bekele, M.K. Clouds-Based Collaborative and Multi-Modal Mixed Reality for Virtual Heritage. *Heritage 2021*, 4, 1447-1459.
- [4] Kidman, B. A Platform for in-Situ Creation of Markerless, Location-Based Augmented Reality Content. Master's Thesis, Dartmouth College, 2023.
- [5] 飛田博章, 渡辺光太郎, 山川美咲, 小島瑛里子. 拡張現実を用いた作品に対するコメントを共有することによる対話型美術鑑賞の支援. 日本・美術による学び学会誌, 第6巻, 第1号.
- [6] Leandro Soares Guedes, Luiz André Marques, and Gabriellen Vitório. "Enhancing interaction and accessibility in museums and exhibitions with Augmented Reality and Screen Readers." Università della Svizzera italiana / Federal Institute of Mato Grosso do Sul.
- [7] 伏田昌弘, 赤羽亭. "画像マーカーベースの AR を用いた音声ガイドの試作." 情報処理学会 インタラクション 2024, IA-16, pp. 253-256, 2024.
- [8] Kyriakou, P. and Hermon, S.: Can I touch this? Using Natural Interaction in a Museum Augmented Reality System, Digital Applications in Archaeology and Cultural Heritage, Vol. 12, e00088 (2018).
- [9] Liu, Y., Spierling, U., Rau, L. and Dörner, R.: Handheld vs. Head-Mounted AR Interaction Patterns for Museums or Guided Tours, Intelligent Technologies for Interactive Entertainment (INTETAIN 2020), LNICST 377, pp. 229-242 (2021).
- [10] Liu, Y., Bitter, J.L. and Spierling, U.: Evaluating Interaction Challenges of Head-Mounted Device-based Augmented Reality Applications for First-time Users at Museums and Exhibitions.
- [11] Ramy Hammady, Minhua Ma, Ziad AL-Kalha, and Carl Strathearn. A framework for constructing and evaluating the role of MR as a holographic virtual guide in museums.

Virtual Reality, Vol. 25, pp. 1-25, 2021.

- [12] 井上道哉, 長澤可也. 綾瀬市埋蔵文化財の VR、 AR コンテンツ化による地域活性化 湘南工科大学紀要, Vol. 55, No. 1, pp. 41-47, 2021.
- [13] Weiting Hou: Augmented Reality Museum Visiting Application based on the Microsoft HoloLens, Journal of Physics: Conference Series, Vol. 1237, 052018, 2019.
- [14] 赤嶺有平: 拡張現実を用いた博物館における双方向メディア型ガイダンスシステムの開発. 科学研究費助成事業 研究成果報告書, 課題番号 19K13045, 2021.
- [15] 星野浩司: AR 型遠隔学習支援システム 「AI Aquarium」 の開発. 九州産業大学芸術学部研究報告, 第 56 卷, pp. 38-41, 2024.
- [16] Robert Lee Seligmann. "Web-based Client for Remote Rendered Virtual Reality". Master's Thesis, Aalto University, 2020.
- [17] Viktor Kelkkanen, Markus Fiedler, and David Lindero. "Synchronous Remote Rendering for VR". International Journal of Computer Games Technology, Vol. 2021, Article ID 6676644, 2021.
- [18] Noor Hammad, Thomas Eiszler, Robert Gazda, John Cartmell, Erik Harpstead, and Jessica Hammer. "V-Light: Leveraging Edge Computing For The Design of Mobile Augmented Reality Games". In Proceedings of the 18th International Conference on the Foundations of Digital Games (FDG '23), 2023.