Master's Thesis

Support for creation of augmented reality game stages based on real room layouts

Ding Yaolong

Supervisor Miyata Kazunori

Division of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Knowledge Science)

February. 2026

Abstract

Augmented reality (AR) games require close spatial alignment between virtual content and real-world environments, which makes stage creation complex, time-consuming, and highly prone to trial-and-error. Unlike conventional game stages authored in fully controlled virtual worlds, AR stages must adapt to unknown indoor layouts, furniture arrangements, walkable areas, and occlusion relationships, while also remaining robust to reconstruction noise and tracking errors on mobile devices. These characteristics significantly increase the cognitive burden of designers and often lead to repeated adjustments during in-situ editing.

To address these challenges, this thesis proposes an AR game stage creation support method based on real room layouts. The method introduces a multi-stage workflow that tightly integrates (1) scene capture and reconstruction, (2) 2D top-down layout planning, (3) automatic 2D-to-AR mapping under geometric constraints, and (4) in-situ AR refinement with immediate gameplay validation. In the proposed workflow, users first externalize macro-stage structure in a low-cognitive-cost 2D representation by arranging predefined game objects on a top-down view of the reconstructed room. The system then maps each object from the 2D layout to a physically plausible 3D placement in the real environment through a projection-based object-wise mapping strategy. Specifically, objects are projected onto the reconstructed environment mesh and refined via local neighborhood search, guided by objectives that balance design fidelity, placement stability, and inter-object overlap suppression. In addition, a low-barrier Trigger–Action authoring mechanism is provided to support interaction design without scripting, together with logic visualization, undo operations, and seamless switching between Editing Mode and Play Mode to encourage rapid iteration.

We evaluate the proposed workflow through a comparative user study with a representative baseline workflow using paper prototyping (hand-drawn sketches for macro-stage planning) followed by in-situ AR editing. A total of 11 participants were recruited and divided into three groups: Group A (2D layout editor + mapping + in-situ AR editing), Group B (paper prototyping + in-situ AR editing), and Group C (third-party playtesting evaluation). Quantitative analysis based on operation logs and production time indicates that the proposed workflow reduces macro-stage planning time and overall production time, and decreases trial-and-error operations during AR editing (e.g., repeated manipulation and deletion). Participants using the proposed workflow also entered Play Mode earlier and more frequently, suggesting a more iterative and validation-driven creation process. Subjective questionnaire results further show higher reuse intention and lower frustration and perceived trial-and-error burden in the proposed condition. Moreover, both creator self-evaluations and third-party playtesting ratings show a favorable trend for stages produced with the proposed workflow in terms of overall quality and immersion.

These results suggest that front-loading macro planning into a 2D layout stage and bridging it to the real environment via automatic mapping can improve the efficiency, usability, and outcome quality of mobile AR game stage creation. The thesis concludes by discussing limitations in reconstruction accuracy, object-wise mapping consistency, and logic expressiveness, and outlines future directions including richer environment understanding, hybrid local–global mapping optimization, and intelligent authoring assistance.

Contents

Chapter 1 Introduction 1
1.1 Research Background 1
1.2 Research Objectives and Contributions 2
1.3 Organization of the Paper 3
Chapter 2 Related Works 4
2.1 AR/VR Game Stage Design Methods 4
2.1.1. In-situ Stage Design 4
2.1.2. Stage Design Based on Abstract Represent 5
2.2 Stage Generation under Real-World Environmental Constraints 6
2.3 AR Stage Implementation in Commercial Game Engines 7
2.4 Scene Reconstruction Techniques 8
2.5 Interior Design Support 10
2.6 Summary 11
Chapter 3 Method 13
3.1 Framework 13
3.2 Scene Reconstruction 13
3.3 2D Layout Design 16
3.4 2D-to-AR Mapping 16
3.4.1. Problem Definition 16
3.4.2. Projection-Based Object-wise Mapping Strategy 17
3.4.3. Objective Function 18
3.4.4. Implementation Details and Default Parameters 22
3.5 Trigger–Action Game Logic Editing 24
3.5.1. Trigger–Action Model 24
3.5.2. Logic Visualization 26
3.5.3. Consistency Between 2D and AR 27
3.6 Play Mode 27
3.7 In-situ AR design 28
Chapter 4 Experimentation 30
4.1 Purpose 30
4.2 Participants 30
4.3 Experimental Conditions and System Implementation 31
4.3.1. Experimental Groups and Procedures 31

4.3.2. Experimental Device 32
4.3.3. System Implementation and Logging 32
4.4 Experimental Tasks 33
4.5 Data Collection Methods 34
4.6 Experimental Results 35
Chapter 5 Evaluation 36
5.1 Evaluation Metrics 36
5.2 Production Efficiency Analysis 37
5.2.1. Total Production Time 37
5.2.2. Macro-Stage Planning Time 38
5.2.3. Operation Count and Operation Density 38
5.3 Operation Behavior and Trial-and-Error Cost Analysis 38
5.3.1. Log-Based Behavior Quantification 38
5.3.2. Comparison of Trial-and-Error Behavior in AR 39
5.3.3. Comparison of Playtest Willingness 39
5.4 Subjective Questionnaire Results (Creator Self-Report) 40
5.5 Stage Outcome Quality and Experience Evaluation 40
5.5.1. Creator Self-Evaluation Results 41
5.5.2. Third-Party Playtest Evaluation Results (Group C) 41
5.6 Summary 41
Chapter 6 Conclusion 43
6.1 Summary of the Research 43
6.2 Limitations 43
6.3 Future Work 44
Acknowledgements 45

List of Figures

Figure 1.1 The poster of Pokémon GO.[1] 1
Figure 1.2 The poster of Minecraft Earth[2]. 1
Figure 2.1 The workflow of "paper prototyping → in-situ AR editing". 5
Figure 2.2 From Scene Graph to AR Scene. 6
Figure 2.3 Constraint-based VR game stage generation adapted from Liu et al.[6] 7
Figure 2.4 The 3DGS reconstruction results.[16] 9
Figure 2.5 The 2DGS reconstruction results.[17] 10
Figure 2.6 The generated results of FlairGPT.[18] 11
Figure 3.1 Framework. 13
Figure 3.2 A standardized capture protocol. 14
Figure 3.3 A reconstruction example. 15
Figure 3.4 The interface of 2D Layout Editor. 16
Figure 3.5 Projection-Based Object-wise Mapping Workflow. 17
Figure 3.6 The mapping results preserve design fidelity. 19
Figure 3.7 Illustration of stability computation. 19
Figure 3.8 Stable placement (a) and unstable placement (b). 20
Figure 3.9 Illustration of AABB-based overlap detection. 21
Figure 3.10 Object-wise Local Mapping Algorithm. 22
Figure 3.11 Principle of On Touch Trigger Implementation. 24
Figure 3.12 The types of Trigger-Action. 25
Figure 3.13 An example of creating a trigger–action. 26
Figure 3.14 The logic visualization. (a)Red arrows indicate Disable relationships, (b)Green arrows indicate Enable relationships. 27
Figure 3.15 The logic visualization in AR. 27
Figure 3.16 UI illustration of switching between Editing Mode and Play Mode... 28
Figure 3.17 In-situ Design Workflow. 29
Figure 4.1 Initial conditions for the three experimental groups. 32
Figure 4.2 Example of an operation log. 33
Figure 4.3 The experimental environment. 34
Figure 4.4 All experimental results related to stage layout. 35
Figure 5.1 Stacked bar chart comparing the distribution of operation types between Group A and Group B. The numbers shown next to each group indicate the number of Play Mode sessions. 39

List of Tables

Table 5.1 Production Time and Operation Statistics per Participant. 37
Table 5.2 Subjective Questionnaire Results (Creator Self-Report). 40
Table 5.3 Comparison of Stage Outcome Quality and Experience Ratings. 40

Chapter 1
Introduction

This chapter introduces the background and motivation of AR game stage creation in real indoor environments. It then states the research objectives and contributions of this work and outlines the organization of the report.

1.1 Research Background

With the increasing computational power of mobile devices and the widespread adoption of spatial sensing technologies such as plane detection, SLAM, and depth estimation, AR games have become an important direction in digital entertainment and interaction research

For example, Pokémon GO[1] became a globally popular and widely discussed title by combining real-world locations with virtual creature-catching gameplay through AR and spatial localization technologies. Microsoft also launched Minecraft Earth,[2] which uses mobile AR and spatial localization to overlay Minecraft blocks, structures, and creatures onto real-world environments. The fact that these popular game IPs have actively released AR-based titles demonstrates the significant potential of the AR entertainment market.

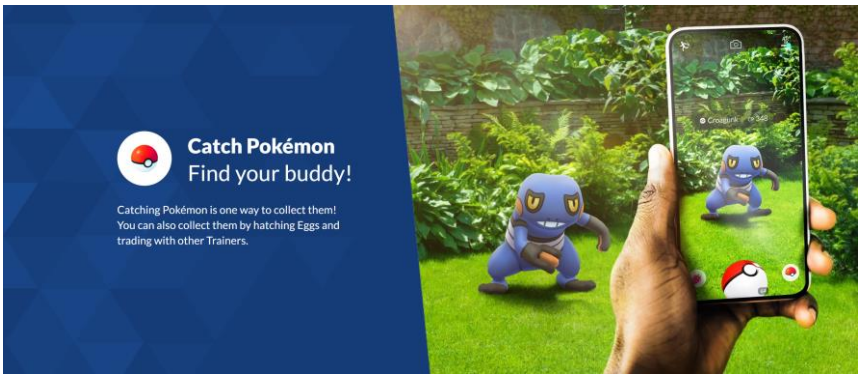


Figure 1.1 The poster of Pokémon GO.[1]

Figure 1.2 The poster of Minecraft Earth[2].

As illustrated in Figure 1.1, the gameplay of Pokémon GO[1] allows players to use their smartphones to capture Pokémon in the real world, as well as to cooperate or compete with other players using these Pokémon, which are presented in an augmented reality (AR) format. As shown in Figure 1.2, Minecraft Earth[2] enables players to place virtual game environments into real-world spaces via AR technology and collaboratively explore these environments together with other players.

Unlike traditional video games based on pre-defined virtual environments or virtual reality (VR) games, AR games require virtual content to be spatially aligned with the real-world environment in which players are situated. Moreover, this alignment must remain consistent under conditions such as user movement, viewpoint changes, and lighting variations in order to achieve immersive spatial interaction experiences.

However, AR game stage design faces challenges that are fundamentally different from those of conventional games. First, real-world environments impose strong constraints and uncertainties—such as furniture placement, walkable areas, and occlusion relationships—making it difficult to coordinate the placement of virtual objects, navigable paths, and interaction logic with physical space. Second, localization errors, reconstruction noise, and incomplete occlusion handling on mobile AR devices often result in visual artifacts such as floating objects, mesh penetration, or inconsistencies between virtual content and the environment. Third, AR stage design commonly relies on first-person in-situ editing. While this approach is intuitive, it makes rapid layout adjustment difficult and leads to high cognitive load and frequent trial-and-error, significantly reducing production efficiency.

Therefore, under the practical constraints of mobile AR environments, how to lower the barrier to AR game stage creation while simultaneously balancing efficient global planning and fine-grained spatial adjustment remains an urgent challenge to be addressed.

1.2 Research Objectives and Contributions

This study aims to propose an AR game stage creation support method based on real room layouts. By combining 2D planar design with in-situ AR editing, we construct a complete stage design workflow for mobile platforms. Within this workflow, designers can complete global stage planning with reduced cognitive load and subsequently verify and refine details in an AR environment, thereby improving the efficiency and controllability of AR game stage creation.

The main contributions of this paper are summarized as follows:

A multi-stage AR game creation workflow for mobile devices

This paper proposes and implements a multi-stage game creation workflow consisting of "scene capture and reconstruction → 2D top-down layout design → automatic 2D-to-3D mapping and adjustment → in-situ AR editing → immediate gameplay validation." By shifting global layout planning to the 2D design step and reserving detailed calibration and experience verification for the in-situ AR editing step, the proposed workflow effectively balances global planning efficiency and local adjustment capability under real-world environmental constraints.

A method for mapping virtual objects from 2D layouts to 3D AR environments

This paper proposes a method that automatically maps virtual objects configured in a 2D planar layout into a 3D AR environment. The method takes the planar positions of virtual objects in the 2D layout as input, determines the placement of each virtual object

in 3D AR space, and refines the mapping results through local spatial search. This process preserves the user's design intent while ensuring placement stability and spatial plausibility in real environments, thereby reducing issues such as floating objects, penetration, and unreasonable overlaps.

Validation through a comparative user study

Using a representative baseline workflow (paper prototyping combined with in-situ AR editing tools) for comparison, a user study is conducted to analyze the proposed method (2D layout editing tools combined with in-situ AR editing tools) in terms of creation efficiency, cognitive load, and overall user experience, thereby validating the effectiveness of the proposed stage creation support approach in practical use.

1.3 Organization of the Paper

This paper is organized into six chapters.

Chapter 2 reviews related work, surveying existing studies on AR/VR game stage design methods, in-situ editing, stage design based on planar or abstract representations, stage generation under real-world environmental constraints, and scene reconstruction techniques, and clarifies the positioning of this research.

Chapter 3 presents the proposed AR game stage creation support method in detail, including the overall system workflow, 2D planar editing approach, automatic 2D-to-3D mapping and optimization methods, as well as in-situ AR editing and game logic editing mechanisms.

Chapter 4 describes the user study design, including the experimental objectives, participant information, experimental procedures, tasks, and data collection methods.

Chapter 5 evaluates and analyzes the experimental results by comparing the proposed method with the baseline workflow in terms of creation efficiency, operational burden, design expressiveness, and final stage outcomes, followed by a discussion.

Finally, Chapter 6 summarizes the contributions of this work, discusses its limitations, and outlines directions for future research.

Chapter 2
Related Works

This chapter reviews prior work related to AR/VR stage design, including in-situ editing, abstract or planar representations for layout planning, and stage generation under real-world constraints. It also summarizes relevant reconstruction techniques and clarifies the research gap addressed by this study.

2.1 AR/VR Game Stage Design Methods

2.1.1. In-situ Stage Design

In-situ stage design refers to an approach in which designers directly place and adjust virtual objects within an AR environment from a first-person perspective. This approach enables immediate feedback in real space and offers clear advantages in terms of spatial consistency and experiential alignment, making it particularly suitable for fine-grained refinement of the final user experience.

Ng et al.[3] proposed a representative in-situ editing system that allows users to construct AR game stages directly from a first-person viewpoint using modalities such as gestures and voice input. While this system emphasizes the precision of in-situ editing, it still relies on a separated workflow for global layout planning, typically following a "paper prototyping → in-situ AR editing" process. In this workflow, users first complete layout planning on paper and then switch to AR for detailed adjustments. Such separation increases the cost of transitioning between representations and often results in considerable trial-and-error during the AR editing phase.

In addition, these systems commonly depend on high-end hardware, such as head-mounted displays equipped with depth sensors, to achieve stable spatial localization. This requirement limits their applicability and scalability on mobile devices.

Overall, while in-situ editing excels at fine adjustment and experiential alignment, it lacks a low-effort planning representation for macro-stage. In this study, the "paper prototyping → in-situ AR editing" workflow is treated as a representative baseline and is used as a comparison condition in the user study to examine whether the proposed approach can reduce trial-and-error during the AR phase and improve overall efficiency.

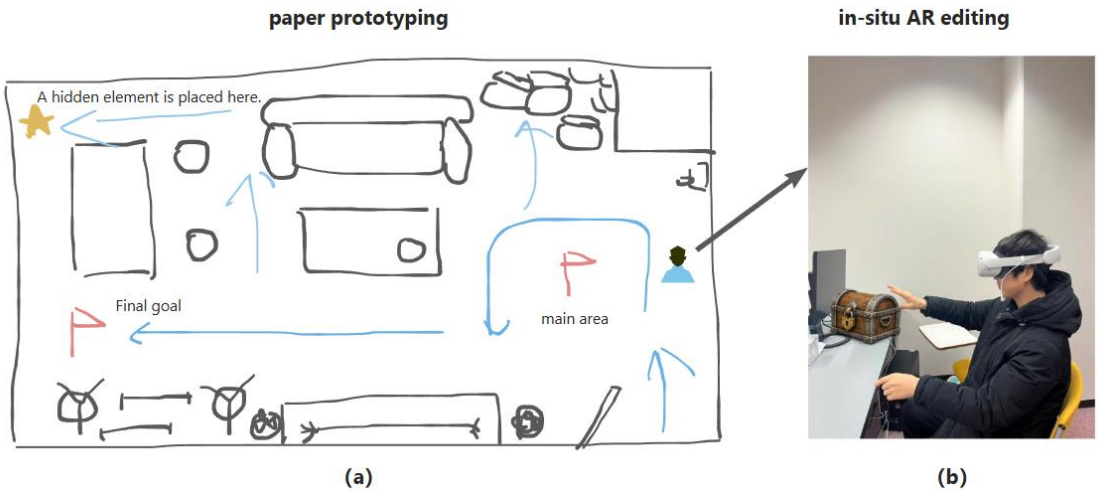


Figure 2.1 The workflow of "paper prototyping → in-situ AR editing".

The user workflow of this type of method is illustrated in Figure 2.1. Typically, users are required to first hand-draw a stage layout according to the surrounding environment (as shown in Figure 2.1(a)), and then perform in-situ editing using devices such as head-mounted displays, placing objects in the game stage from a first-person perspective (as shown in Figure 2.1(b)).

2.1.2. Stage Design Based on Abstract Represent

Another line of research adopts planar representations or abstract structures, such as top-down layouts or scene graphs, to design stages, and then maps these representations into AR space. These approaches support macro-stage structural planning and help designers reason about overall layouts, but they typically lack direct support for real-world geometric details and first-person experiential verification.

For example, Liu et al.[4] abstract both the stage and the real environment into graph structures, where nodes represent objects and edges represent spatial relationships. By performing topological matching, their method enables rapid mapping of designed stages into real environments. While such approaches are effective for global structure matching, they require users to understand abstract graph representations, leading to higher learning costs. Moreover, they provide limited control over absolute object positions and fine-grained spatial layouts.

Similarly, Tahara et al.[5] proposed Retargetable AR, a framework that represents both AR content and real environments using abstract 3D scene graphs constructed from geometric and semantic relationships rather than explicit coordinate transformations. By matching an AR scene graph with a scene graph generated from RGB-D observations of the real environment, their method enables context-aware content arrangement across different indoor scenes. While this approach effectively supports semantic consistency and context adaptation, it focuses primarily on semantic alignment and interaction coherence, rather than interactive stage authoring or iterative layout refinement from a designer's perspective.

Overall, planar or abstract representations are well suited for macro-stage planning but

are often difficult to directly integrate with real-world geometric constraints and in-situ editing workflows. In contrast, this study adopts an intuitive 2D top-down layout as an intermediate representation and tightly integrates it with reconstructed environment geometry, enabling a continuous workflow from global planning to geometry-constrained 3D placement and first-person validation.

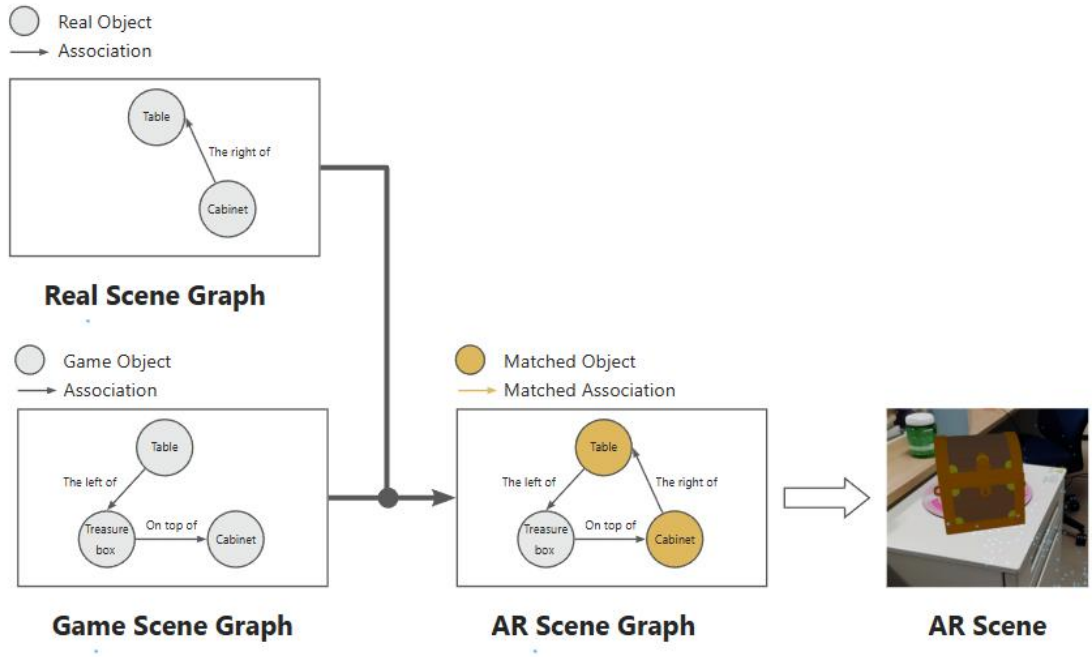


Figure 2.2 From Scene Graph to AR Scene.

Figure 2.2 illustrates a graph-based AR scene construction process. Spatial relationships among virtual objects are first modeled using an abstract game scene graph.

Meanwhile, the real environment is first analyzed in 3D by extracting semantic object candidates and their oriented bounding boxes from RGB-D point clouds. Object categories (e.g., tables and cabinets) are then inferred based on geometric cues such as orientation, size, and height, and organized into a real scene graph together with geometry-based spatial relationships.

The two scene graphs are then matched to instantiate virtual objects in the real world and construct the final AR scene.

2.2 Stage Generation under Real-World Environmental Constraints

Some studies attempt to leverage structural information from real-world environments to improve the adaptability of virtual game stages through automatic generation or constraint-based optimization. For example, Liu et al.[6] proposed a VR-based stage generation approach in which users first annotate obstacle regions within a real environment, after which the system automatically generates a modular stage layout by optimizing a cost function that encodes environmental constraints and layout feasibility. This line of work emphasizes environmental constraint modeling and the efficiency of automatic generation, and can ensure the feasibility and safety of the generated layouts to a certain extent.

In addition, Sun et al.[7] proposed Mapping Virtual and Physical Reality, which focuses on the geometric correspondence between virtual and physical spaces to support immersive experiences that allow natural walking in real environments. Their method constructs a mapping between planar layouts of virtual scenes and physical spaces that minimizes angular and distance distortions, enabling large virtual environments to be spatially remapped to fit within smaller physical spaces while avoiding physically implausible intersections between virtual objects. This work provides important insights into walkable path planning and coordination between virtual and physical spaces, and demonstrates its potential applications in gaming and immersive navigation scenarios.

Despite their strengths in environmental constraint modeling and automatic generation, these approaches are largely driven by automatic optimization processes with limited direct user intervention. As a result, designers are given relatively limited direct control over stage structure and interaction details, and it is difficult to support frequent interactive modifications and explicit expression of creative intent. In highly creative stage design tasks, excessive reliance on automatic generation may reduce designers' subjective control and exploratory freedom.

Therefore, this study adopts a hybrid strategy of "user-driven layout design with system-assisted mapping." In this approach, designers first define the macro-stage structure in a low-cognitive-cost representation, after which the system performs spatial mapping and placement optimization under real-world environmental constraints. This strategy introduces environmental constraints while preserving design freedom and controllability to the greatest possible extent.

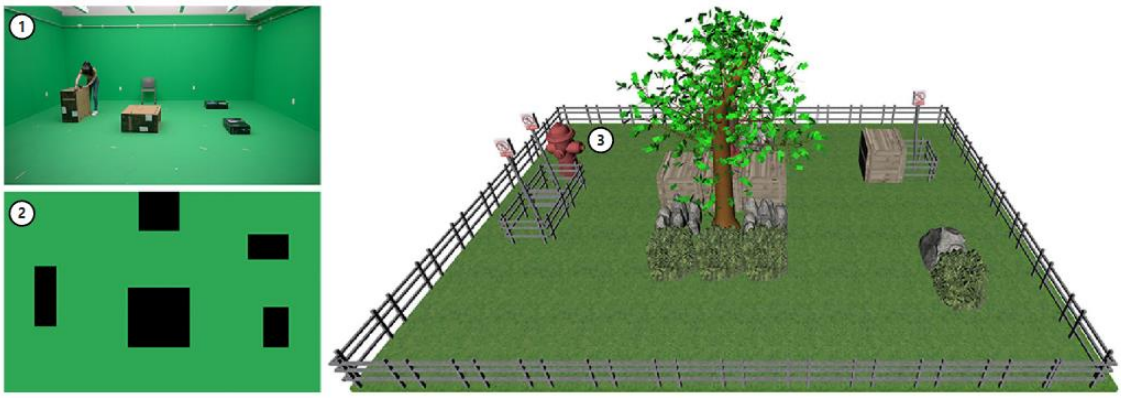


Figure 2.3 Constraint-based VR game stage generation adapted from Liu et al.[6]

As shown in Figure 2.3 ①, a real room is prepared using physical proxies such as cardboard boxes and chairs to represent non-navigable regions.

Users then define the overall navigable area and obstacle regions using a VR controller, resulting in a 2D spatial layout as shown in Figure 2.3 ②.

Based on the user-defined constraints, a game stage that satisfies the environmental and navigational constraints is automatically generated, as illustrated in Figure 2.3 ③.

2.3 AR Stage Implementation in Commercial Game Engines

For professional developers, a common practice is to develop AR games using Unity[8]

and AR Foundation.[9] This approach provides comprehensive functionality and strong cross-platform support, enabling the use of plane detection, spatial anchors for stabilizing virtual objects in real-world locations, and occlusion handling. However, the development and stage creation process requires substantial engineering expertise and familiarity with complex tool chains, making it unsuitable for users without AR or game development experience.

The system proposed in this paper is also implemented using Unity and AR Foundation, in combination with the third-party spatial perception plugin EasyAR.[10] The key difference is that this study provides a simplified, task-oriented, and integrated interface specifically designed for stage creation, along with a multi-stage editing workflow, thereby lowering the barrier to use.

2.4 Scene Reconstruction Techniques

Scene reconstruction aims to estimate scene geometry and appearance from video images or depth data and output a three-dimensional representation that can be used in virtual environments. Depending on the underlying representation, existing scene reconstruction methods can be broadly categorized into point-cloud-based methods, mesh-based methods, implicit representations, and Gaussian-based representations.

Point-Cloud-Based Reconstruction

Point-cloud-based methods represent scenes as discrete sets of points in three-dimensional space, typically recovered through multi-view geometric constraints. Classical approaches based on Structure-from-Motion (SfM)[11] and Multi-View Stereo (MVS)[12] reconstruct sparse to dense point clouds from multi-view RGB images. And COLMAP[13] is a widely used open-source system that provides a complete implementation of the SfM–MVS pipeline, integrating camera pose estimation, sparse reconstruction, and dense multi-view stereo.

These methods are widely adopted due to their geometric accuracy and mature engineering pipelines. However, point clouds lack explicit surface connectivity, which limits their direct applicability to tasks such as collision detection, occlusion reasoning, and stable object placement. As a result, additional surface reconstruction or meshing steps are often required before integration into AR systems or game engines.

Mesh-Based Reconstruction

Mesh-based methods explicitly represent scene geometry using surface meshes, providing continuous surface information that is well suited for physical interaction and spatial reasoning. Representative approaches include depth-fusion-based systems such as KinectFusion[14], which integrate multiple depth frames into a Truncated Signed Distance Function (TSDF) volume and extract mesh surfaces from it.

Mesh representations offer strong compatibility with AR/VR systems and game engines, enabling direct use in collision detection, occlusion handling, and physics simulation. However, these methods often rely on depth sensors or high-quality depth estimation, and their reconstruction quality may degrade when only RGB inputs are available.

Implicit Representation Methods

Recent years have seen significant advances in implicit scene representations, most notably Neural Radiance Fields (NeRF)[15]. NeRF models scene geometry and

appearance as a continuous volumetric function using neural networks and synthesizes novel views via volume rendering.

Implicit representations achieve high visual fidelity and strong view consistency, making them particularly effective for photorealistic rendering. However, the reconstructed scenes are stored as implicit functions rather than explicit geometry. For AR applications that require geometric reasoning—such as collision detection, spatial constraints, or object placement—additional surface extraction or approximation steps are typically required, increasing system complexity.

Gaussian-Based Reconstruction (3DGS)

3D Gaussian Splatting (3DGS)[16] represents scenes using a collection of three-dimensional Gaussian primitives that jointly encode geometry and appearance. This representation achieves a favorable balance between rendering quality and computational efficiency, enabling real-time or near real-time rendering while maintaining high visual quality.

Compared with implicit methods, 3DGS significantly reduces training and rendering costs. However, similar to NeRF, its primary focus lies in view synthesis and rendering. The representation does not directly provide explicit surface geometry, which limits its applicability for geometry-constrained AR editing and spatial reasoning tasks.

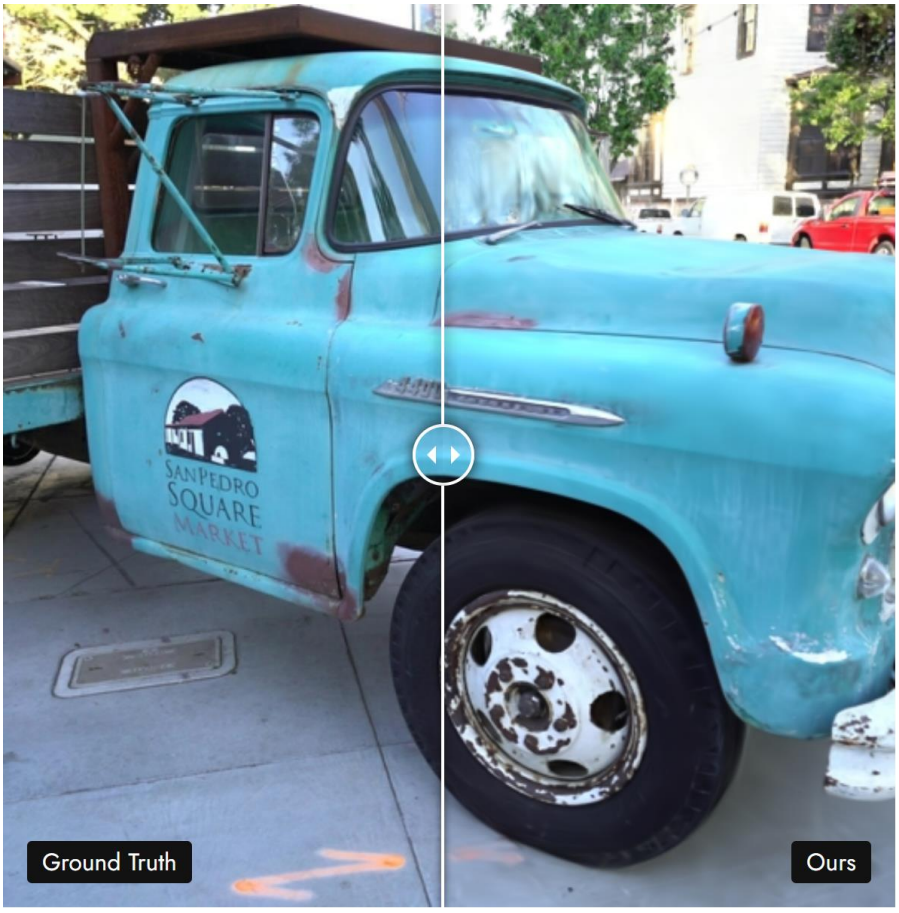


Figure 2.4 The 3DGS reconstruction results.[16]

Figure 2.4 shows the high-quality results of 3D Gaussian Splatting (3DGS). The left

side presents photographs of the captured real-world scene (ground truth), while the right side shows the reconstructed results.

2D Gaussian Splatting (2DGS)

Building upon Gaussian-based representations, 2D Gaussian Splatting (2DGS)[17] models scenes as a set of oriented two-dimensional Gaussian surface elements (surfels) and supports conversion from Gaussian representations to mesh-based geometry.

Compared with point clouds or direct Gaussian rendering, the mesh outputs derived from 2DGS offer improved geometric structure and engineering compatibility. Mesh representations can be directly used for collision detection, occlusion reasoning, and spatial constraint computation in game engines and AR systems. Therefore, this study adopts 2DGS as the scene reconstruction approach, providing a stable geometric foundation for subsequent stage mapping, object placement optimization, and in-situ AR editing.

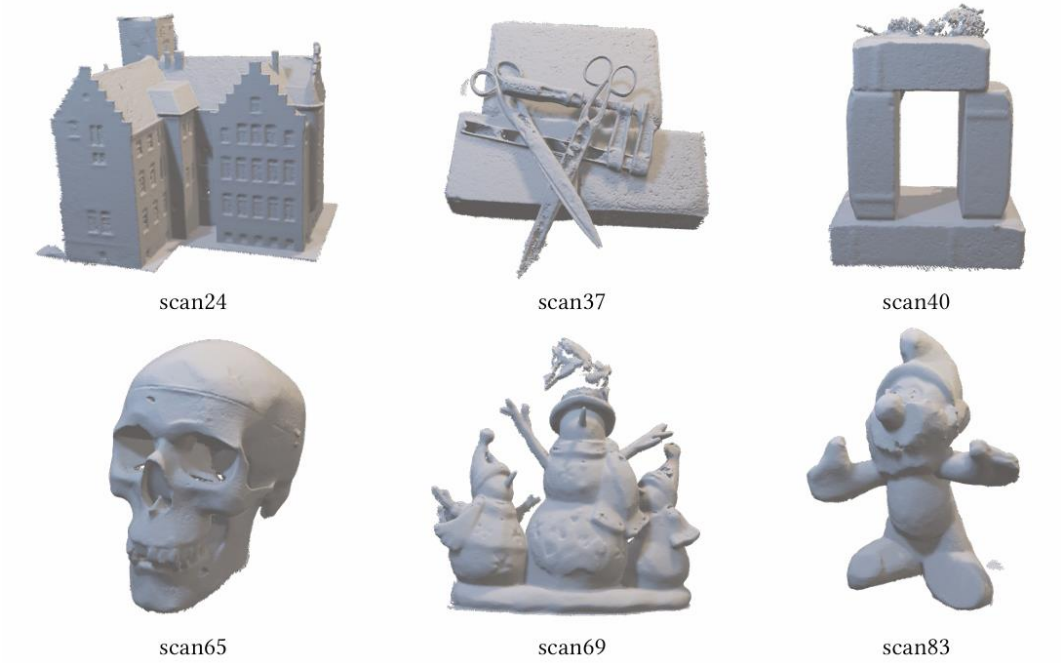


Figure 2.5 The 2DGS reconstruction results.[17]

2.5 Interior Design Support

Beyond game stage design, recent studies have explored general indoor scene generation and creation support systems, primarily targeting interior design scenarios. These works focus on reducing the barrier for non-expert users, improving layout rationality, and enabling high-level intent expression through abstract representations such as natural language or scene graphs.

Representative approaches in this line of research leverage large language models to translate user intent into structured spatial descriptions and automatically generate complete indoor scenes. For example, Littlefair et al. proposed FlairGPT[18], which uses natural language input to plan room layouts and furniture selection, and generates indoor scenes from a predefined object library. More recently, Aguina-Kang et al.[19] introduced

an open-universe indoor scene generation framework that synthesizes layout programs using large language models and solves the resulting constraint satisfaction problem to place objects retrieved from large, uncurated mesh databases. Similarly, Çelen et al.[20] proposed I-Design, a personalized interior design system in which multiple LLM-based agents collaboratively reason over user input to construct scene graph representations and determine feasible object placements within a 3D scene.

Collectively, these methods demonstrate the effectiveness of LLM-based approaches in supporting high-level layout planning, abstract concept interpretation, and automatic indoor scene generation. However, they are primarily designed for decorative or functional interior design tasks and typically do not address gameplay-oriented requirements, such as player path planning, interaction triggers, or precise spatial alignment with real-world geometry. As a result, they are difficult to apply directly to AR game stage design, which demands accurate geometric consistency, controllable interaction logic, and first-person experiential validation.

In contrast, this study focuses on AR game stage creation under real-room environmental constraints. By combining 2D planar layout planning with in-situ AR editing, the proposed approach supports the co-optimization of game structure design and spatial consistency, while preserving direct designer control over gameplay flow and interaction structure.

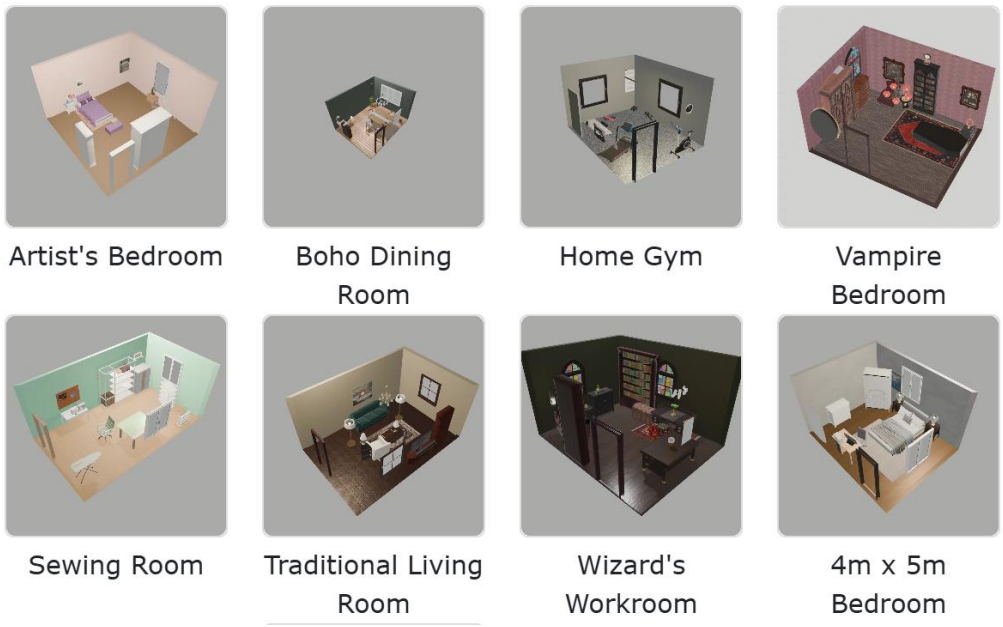


Figure 2.6 The generated results of FlairGPT.[18]

2.6 Summary

In summary, existing studies have proposed various AR stage design approaches from the perspectives of in-situ editing, abstract representation, and generation. However, there is still a lack of a unified workflow that simultaneously supports:

- low-cost macro-stage layout expression,
- consistency with real-world geometric constraints,
- fine-grained adjustment and rapid validation.

The proposed approach allows designers to first plan the overall stage structure using a user-driven 2D top-down layout, then automatically place the layout into the real 3D environment, and finally refine object positions through in-situ AR editing. Specifically, this study constructs a multi-stage workflow consisting of "scene reconstruction → 2D layout design → automatic mapping → in-situ AR editing," enabling designers to complete global planning in a low-cost representation before entering AR for limited but necessary refinements. This approach reduces trial-and-error during the AR phase and improves overall creation efficiency.

In the experimental design, the commonly used workflow of "paper prototyping → in-situ AR editing," as seen in the work of Ng et al., is adopted as a representative baseline. This baseline is compared with the proposed workflow to evaluate differences in efficiency, cognitive burden, and user experience.

Chapter 3
Method

This chapter presents the proposed AR stage creation support method and its overall multi-stage workflow. It describes the system components in detail, including 2D layout design, 2D-to-AR mapping with local optimization, in-situ AR editing, and Trigger–Action logic authoring with play-mode validation.

3.1 Framework

The proposed system consists of four main steps, as illustrated in Figure 3.1:

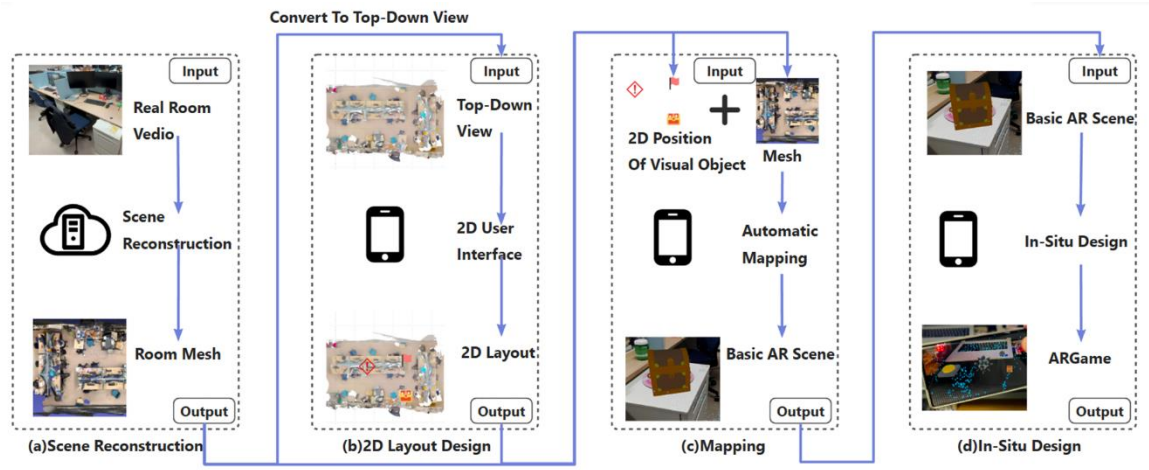


Figure 3.1 Framework.

Scene Reconstruction

Indoor videos are captured using a mobile device, and a scene mesh is generated from the input video using 2DGS.

2D Layout Design

A top-down view is generated based on the reconstructed mesh. In this step, users quickly define the overall stage structure logic by dragging object icons within the 2D layout interface.

Mapping

Objects placed in the 2D layout are projected one by one onto the 3D scene mesh. Their positions are then refined through local search within a limited neighborhood to obtain stable and spatially plausible placements in the AR environment.

In-situ Design

Users fine-tune object position, rotation, and scale from a first-person AR perspective, and can directly run the game at any time to test the stage through gameplay.

3.2 Scene Reconstruction

The system captures video data of the real environment using a mobile device and

reconstructs the scene using 2DGS. 2DGS represents the scene as a set of oriented surface elements in three-dimensional space and supports further conversion into a mesh-based geometric representation, allowing the reconstruction results to be directly used for spatial computations in game engines.

Compared with point clouds or direct Gaussian-based rendering representations, mesh representations offer stronger engineering compatibility in game engines such as Unity. In particular, they can be directly utilized for several operations that are critical to stage creation, including:

- evaluating the stability of virtual object placement (i.e., whether a supporting surface exists);
- detecting penetration and overlap between virtual objects and the environment;
- analyzing occlusion relationships and navigability based on geometric structure.

Therefore, converting reconstructed scenes into mesh representations not only improves visual consistency but also provides essential geometric constraints for subsequent 2D-to-AR mapping and placement optimization.

From the user's perspective, the workflow requires only capturing and uploading a video of the indoor environment using a mobile device. The reconstruction process itself is performed offline on a PC server, without requiring users to have prior knowledge of 3D modeling or parameter tuning.

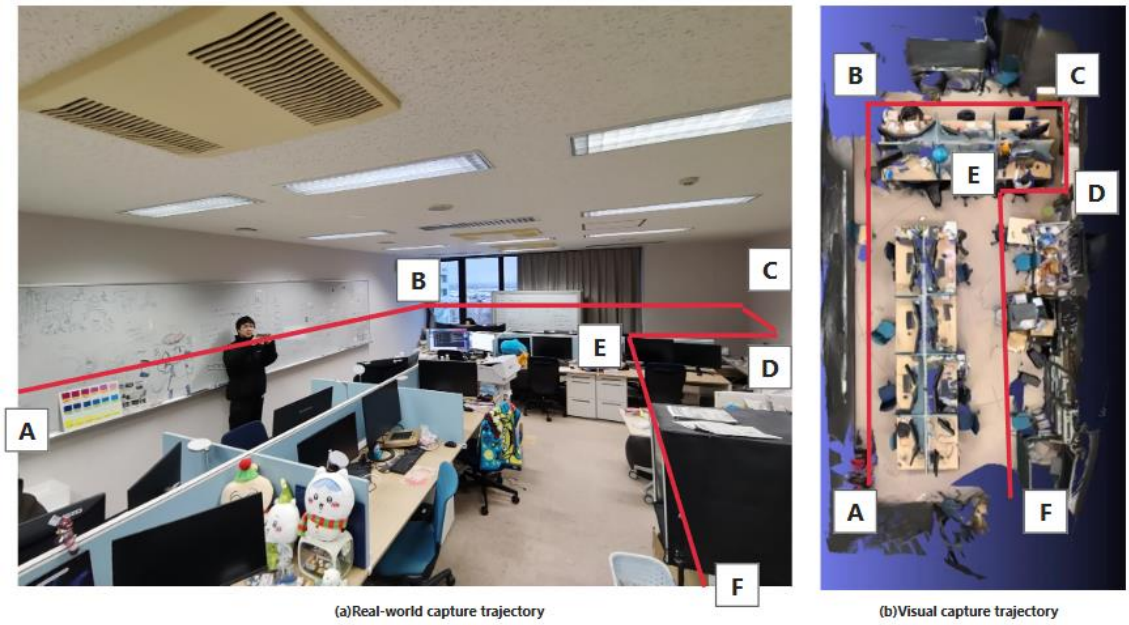


Figure 3.2 A standardized capture protocol.

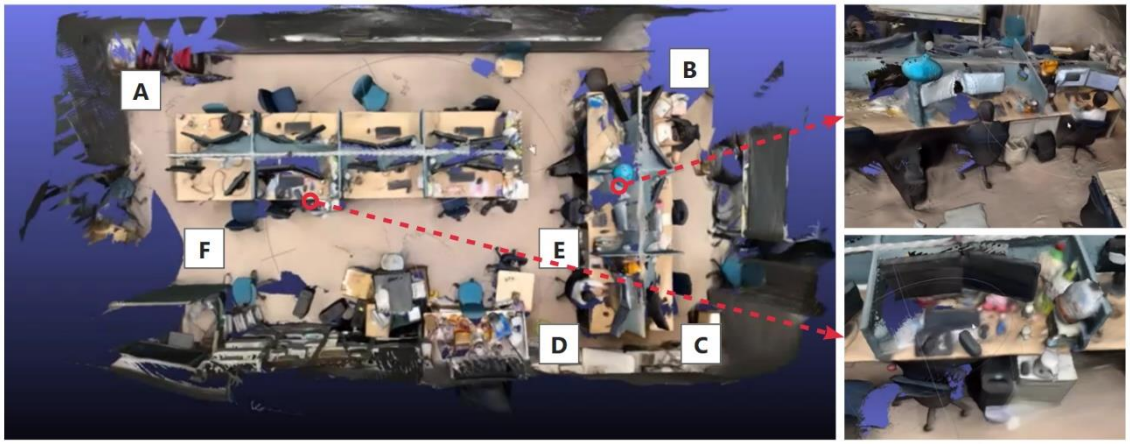


Figure 3.3 A reconstruction example.

In the experimental setting, an indoor space of approximately 78 m² was captured using a mobile device at a resolution of 1080p and a frame rate of 30 fps. Video acquisition followed a standardized capture protocol, which was developed through iterative trial-and-error during our experiments and informed by practical recommendations reported in public 2DGS GitHub repositories.[21]

Specifically, as shown in Figure 3.2(a), the user held the mobile device at approximately shoulder height and walked along the perimeter of the room in a clockwise manner, following the sequence from A to F. During the capture process, the user kept their back close to the surrounding walls while consistently orienting the camera toward the center of the space, as indicated by the red arrows. A slow and steady walking pace was maintained to ensure uniform visual coverage and stable input for scene reconstruction. Figure 3.2(b) shows the same capture trajectory mapped onto the reconstructed scene, where the labeled positions (A–F) correspond to those in the real-world capture.

The 2DGS processing was performed on a machine equipped with an RTX 3080 GPU, and the reconstruction and mesh generation were completed in approximately 30 minutes. Figure 3.3 presents the reconstruction results produced by the 2DGS method reproduced in this study, following the standard processing pipeline.

3.3 2D Layout Design

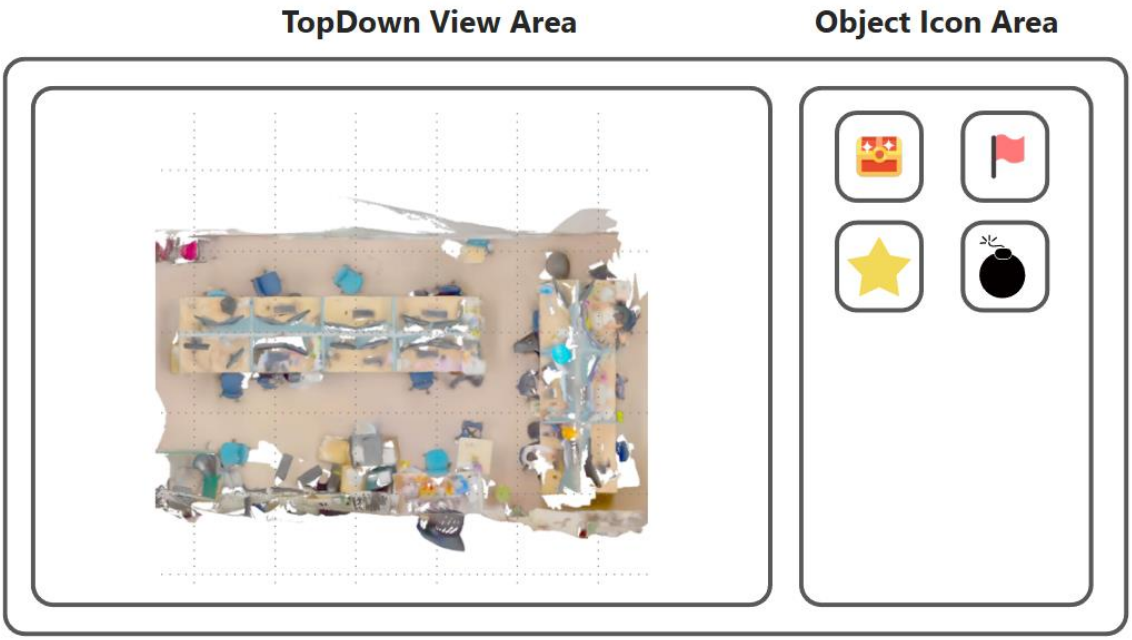


Figure 3.4 The interface of 2D Layout Editor.

The system provides a 2D game stage editor based on a top-down view, as shown in Figure 3.4. The top-down view is generated from the reconstructed scene mesh and serves as a planar reference for stage layout editing.

As illustrated in the interface, the editor is mainly divided into two areas: the Top-down View Area on the left and the Object Icon Area on the right. Users select predefined objects (e.g., chests, flag, stars, bombs) from the Object Icon Area and place them into the Top-down View Area via drag-and-drop interactions, allowing them to quickly define the overall structure of the stage.

The objectives of the 2D planar design step are to:

- rapidly complete macro-stage layout planning;
- clarify the relative spatial relationships among objects;
- reduce the cognitive burden associated with early-stage ideation by avoiding starting from a blank canvas.

During this step, users are not required to specify object height or the exact supporting surface (e.g., floor or tabletop). Such information is automatically inferred and completed by the system during the subsequent mapping step based on the reconstructed scene mesh.

3.4 2D-to-AR Mapping

3.4.1. Problem Definition

Given the planar position of the i-th object in the 2D layout, denoted as (x_i^{2D}, y_i^{2D}), the goal is to determine its final placement pose in the three-dimensional space represented by the reconstructed environment mesh. In particular, the mapping process

aims to determine the object's vertical position (z-axis) and supporting surface such that the resulting placement satisfies the following requirements:

Preservation of design intent: the spatial structure and relative relationships expressed in the 2D layout should be maintained as closely as possible;

Placement stability: objects should not float and should have sufficient physical support from the environment;

Reduction of unreasonable overlap: penetration and occlusion conflicts with the environment or with other virtual objects should be minimized.

3.4.2. Projection-Based Object-wise Mapping Strategy

To avoid the computational complexity and instability associated with high-dimensional global optimization, we adopt an object-wise local mapping strategy. In this approach, each virtual object in the 2D layout is mapped and optimized independently, enabling efficient and stable placement in the AR environment.

For the i-th object in the 2D layout, the system performs the following steps:

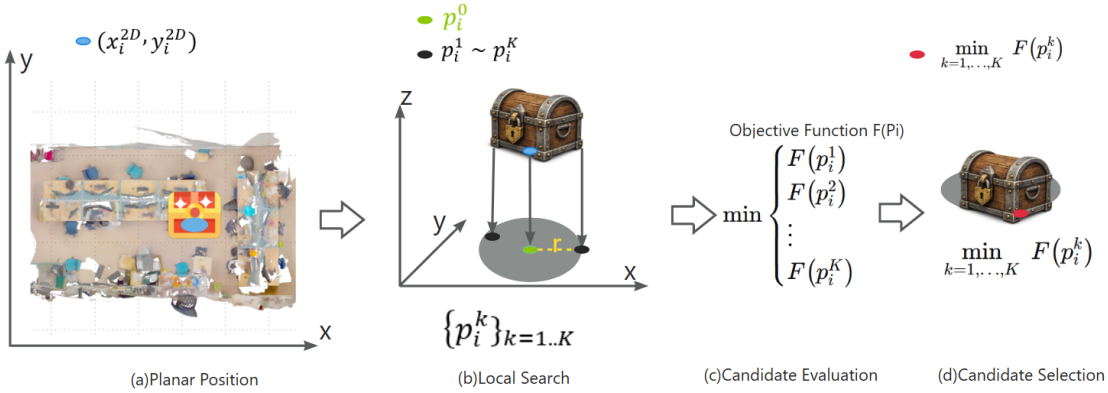


Figure 3.5 Projection-Based Object-wise Mapping Workflow.

Initial Projection

Starting from the planar position (x_i^{2D}, y_i^{2D})in the 2D layout (as shown in Figure 3.5(a)), a ray is cast in the three-dimensional scene along the negative z-axis of the world coordinate system. The ray is intersected with the reconstructed environment mesh to obtain an initial placement point p_i^0. This point represents a direct projection of the 2D layout into 3D space and serves as a reference that preserves the user's original design intent.

Local Search Region Definition

A local search region with radius r is defined around the initial point p_i^0. Within this region, a finite set of candidate placement points $\{p_i^k\}_{k=1..K}$is generated (as shown in Figure 3.5(b)). This local region allows the system to compensate for reconstruction noise or local geometric irregularities while avoiding unconstrained global search.

Candidate Evaluation and Selection

For each candidate point p_i^k, the system evaluates an objective function $F(p_i^k)$to

assess the suitability of the location as a placement point for the object (as shown in Figure 3.5(c)). Specifically, the evaluation considers whether the object is reliably supported by the environment surface, whether it exhibits obvious interpenetration with other geometric structures, and the degree of deviation from the original position defined by the user in the 2D layout. The candidate with the best overall evaluation is selected as the final placement position p_i^* of the object in the 3D AR environment.

Here, r denotes the radius of the local search region, and K denotes the maximum number of candidate points. The specific implementation of the objective function $F(p)$ will be described in detail in Section 3.4.3. Candidate points are generated using a regular sampling strategy, which provides predictable and stable performance suitable for mobile AR scenarios while maintaining placement accuracy.

3.4.3. Objective Function

During the local search step, multiple evaluation objectives are defined to assess candidate placement positions. To balance user design intent and real-world environmental constraints, we introduce three types of evaluation objectives and combine them into a weighted objective function.

(1) Design Fidelity

The design fidelity objective measures the deviation of a virtual object's final placement in three-dimensional space from its original position in the 2D planar layout, with the goal of preserving the layout structure and relative spatial relationships expressed during the 2D design.

Specifically, let (x_i^{3D}, y_i^{3D}) denote the projection of the final placement position of object i onto the horizontal plane in 3D space, and let (x_i^{2D}, y_i^{2D}) denote its original position in the 2D layout, where i=1,…,N and N is the total number of objects to be placed. The design fidelity term is defined as:

$$F_1 = \sum_{i=1}^{N} \| (x_i^{3D}, y_i^{3D}) - (x_i^{2D}, y_i^{2D}) \|^2$$

This term encourages the mapped positions to remain close to the original 2D layout in the horizontal plane, thereby reducing distortion of the overall layout structure caused by automatic mapping.

Figure 3.6 The mapping results preserve design fidelity.

(2) Stability

The stability objective aims to reduce cases where virtual objects are floating, partially supported, or supported only by a small number of contact points, ensuring that objects are placed in a physically plausible and stable manner in the real environment.

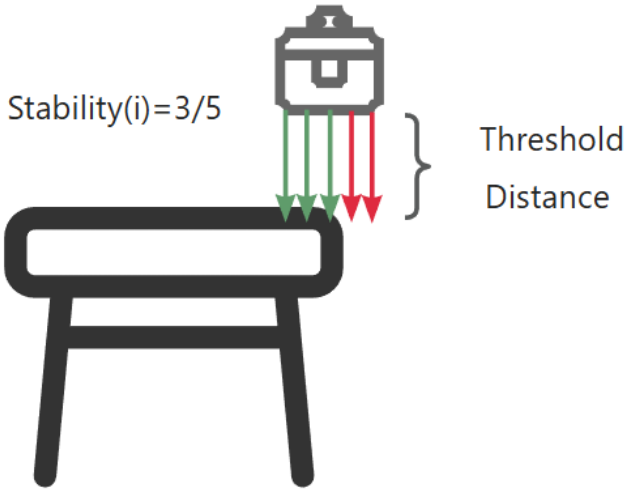


Figure 3.7 Illustration of stability computation.

As illustrated in Figure 3.7, stability is evaluated by sampling points on the bottom surface of each object and estimating the proportion of the surface that is supported by the environment mesh.

Specifically, for object i, M sampling points are uniformly distributed over its bottom surface. From each sampling point, a short ray is cast downward along the negative z-axis. If the ray intersects the environment mesh within a predefined support threshold distance, the corresponding sampling point is considered supported; otherwise, it is treated as unsupported, as shown in the figure.

Let $Stability_i$ denote the proportion of supported sampling points among all M samples on object i. The unsupported proportion is then defined as:

$$UnfitArea(i) = 1 - Stability_i.$$

Based on this definition, the stability objective is formulated by aggregating the unsupported proportions over all objects:

$$F_2 = \sum_{i=1}^{N} UnfitArea\ (i).$$

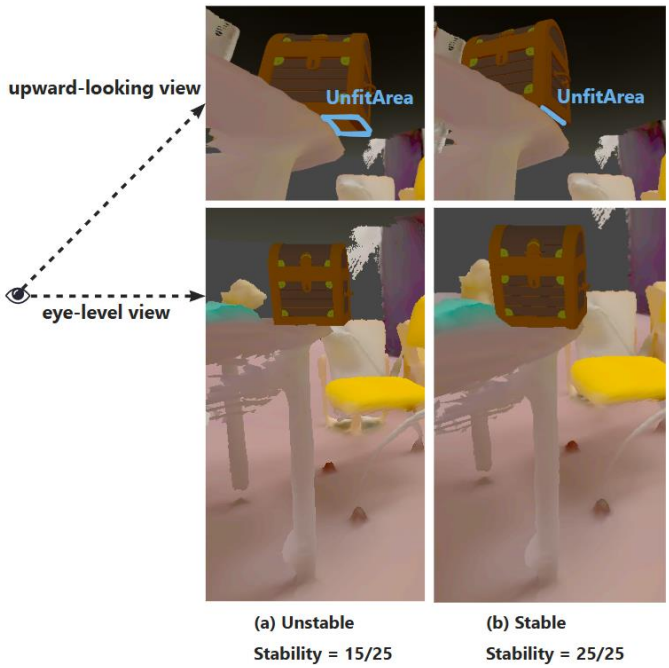


Figure 3.8 Stable placement (a) and unstable placement (b).

Figure 3.8 illustrates stable and unstable object placements observed from eye-level and low-angle viewpoints. Objects with a larger UnfitArea are regarded as unstable, whereas objects with a smaller UnfitArea are considered stable. In the figure, the regions highlighted in blue indicate areas that are classified as UnfitArea, i.e., portions of the object's bottom surface that are not sufficiently supported by the environment. The objective of the stability term is therefore to identify object placements with minimal UnfitArea, ensuring physically plausible and stable placement in the real environment.

(3) Non-overlap

The non-overlap constraint aims to reduce unreasonable interpenetration between virtual objects, thereby avoiding severe penetration artifacts or visually implausible occlusions in the scene.

To ensure computational efficiency, instead of performing exact volumetric intersection tests, we approximate inter-object interference by explicitly measuring the degree of overlap between object pairs.

This overlap measurement serves as a penalty term, where a larger value indicates more severe interpenetration and should therefore be minimized during optimization.

Based on this formulation, the non-overlap objective is defined as the aggregation of pairwise overlap penalties:

$$F_3 = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \text{OverlapPenalty}(i, j),$$

Where N denotes the total number of virtual objects in the scene.

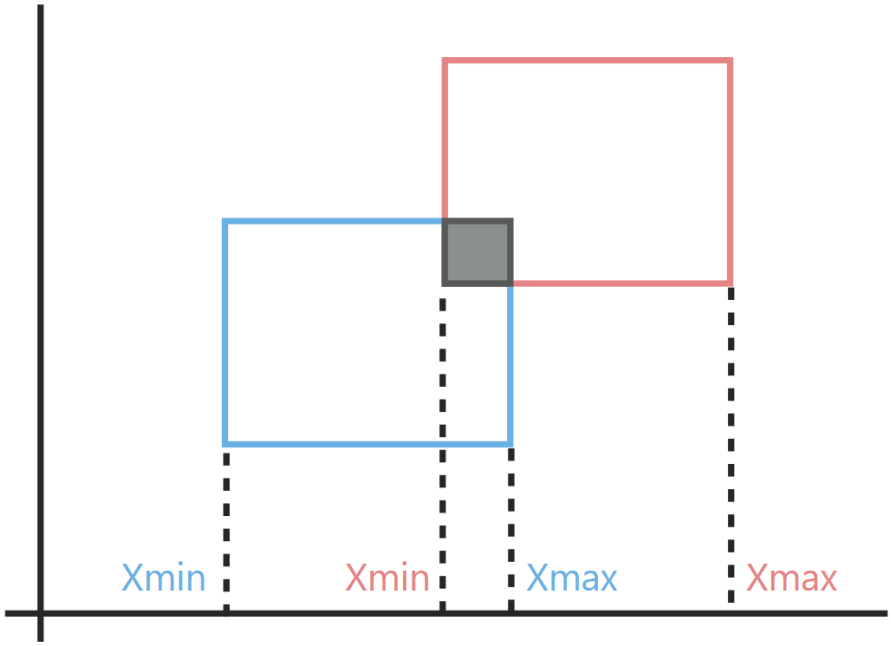


Figure 3.9 Illustration of AABB-based overlap detection.

Figure 3.9 illustrates the object-level overlap detection based on axis-aligned bounding boxes (AABBs).

In this study, each object's geometric extent is conservatively approximated using an AABB, and inter-object overlap is estimated by testing whether the corresponding AABBs intersect in three-dimensional space. When two bounding boxes overlap, a penalty value proportional to the degree of overlap is assigned, serving as a rough approximation of inter-object interpenetration rather than an exact geometric collision measure.

This AABB-based formulation offers the advantages of simple implementation and high computational efficiency. Although it does not capture the exact overlap of complex or non-box-shaped geometries, the resulting penalty is sufficient to suppress visually noticeable interpenetration during layout optimization. By avoiding expensive exact collision or intersection tests, this approach is well suited for interactive or real-time scene layout optimization tasks.

Combined Objective Function

The final evaluation function is defined as a weighted sum of the three objectives:

$$F = \alpha_1 F_1 + \alpha_2 F_2 + \alpha_3 F_3$$

Where α_1, α_2, and α_3 are weighting parameters that balance design fidelity, placement stability, and non-overlap, respectively.

Algorithm 1: Object-wise Local Mapping

```
▸ O : set of 2D objects with planar positions
▸ M : reconstructed environment mesh
▸ r : local search radius
▸ K : number of candidate samples
▸ F : weighted objective function

S^{3D} ← ∅                                          ▸ initialize output layout
for o_i ∈ O do                                      ▸ iterate over objects
    p_i^0 ← RaycastDown((x_i^{2D}, y_i^{2D}), M)    ▸ initial 2D-to-3D projection
    P_i ← SampleCandidates(p_i^0, r, K)             ▸ local candidate sampling
    for p_i^k ∈ P_i do                              ▸ evaluate candidates
        F_1 ← DesignFidelity(p_i^k)                 ▸ horizontal deviation
        F_2 ← Stability(p_i^k)                      ▸ surface support
        F_3 ← NonOverlap(p_i^k)                     ▸ collision penalty
        F(p_i^k) ← α_1·F_1 + α_2·F_2 + α_3·F_3      ▸ weighted objective
    end for
    p_i^* ← argmin_{p ∈ P_i} F(p)                   ▸ select optimal position
    position(o_i) ← p_i^*                           ▸ assign 3D translation
    S^{3D} ← S^{3D} ∪ {o_i}
end for
return S^{3D}
```

Figure 3.10 Object-wise Local Mapping Algorithm.

3.4.4. Implementation Details and Default Parameters

In our implementation, the object-wise local mapping adopts a fixed set of default parameters to ensure stable behavior and predictable runtime on mobile devices.

Local search

For each object i, the initial projection point p_i^0 is obtained by casting a ray along the negative world z-axis onto the reconstructed scene mesh. A local search region is defined as a horizontal neighborhood centered at p_i^0 with radius $r = 0.25$m. Candidate points are generated via regular grid sampling on the horizontal plane using a 9×9grid ($K = 81$) with spacing $\Delta = 0.0625$ m, clipped to the circular region of radius r. For each candidate position (x, y), the final height z is determined by a downward ray–mesh intersection to identify the supporting surface.

Stability evaluation

To evaluate placement stability, sampling points are distributed over the supporting surface of each object, defined as the portion of the object that is expected to be in contact with the environment under gravity. A fixed sampling density of 25 points per square meter is used, and the total number of sampling points for an object is computed based on the area of its supporting surface. The sampling points are generated in the object's local coordinate system and transformed into world space.

From each sampling point, a short ray is cast downward toward the environment. If the ray intersects the scene mesh within a predefined support threshold distance (0.02 m), the corresponding sampling point is considered supported; otherwise, it is treated as unsupported. The stability of an object placement is then assessed by estimating the proportion of supported sampling points across the entire supporting surface.

Let M denote the total number of sampling points distributed on the supporting surface of object i, and let $N_i^{\text{supported}}$ denote the number of sampling points whose downward rays intersect the scene mesh within the support threshold. The stability score is defined as:

$$\text{Stability(i)} = \frac{N_i^{\text{supported}}}{M}.$$

A higher proportion indicates that the object is well supported by the environment, while a lower proportion suggests that the object is partially floating or supported only at limited contact regions. The complementary proportion of unsupported sampling points is used to represent unstable or insufficiently supported areas of the object.

This area-based sampling strategy provides a consistent measure of support across objects of different sizes while maintaining predictable computational cost, making it suitable for interactive or real-time placement optimization.

Overlap approximation

Object–object overlap is approximated using axis-aligned bounding box (AABB) intersection tests. For each pair of objects *i* and *j*, their geometric extents are approximated by AABBs, and overlap is detected by testing whether the corresponding bounding boxes intersect in three-dimensional space. When an intersection occurs, the overlap ratio is estimated as the intersection volume divided by the volume of the smaller AABB, so as to normalize the overlap with respect to object scale and emphasize cases where a smaller object is largely intruded, providing an efficient approximation of inter-object overlap.

This AABB-based overlap measure is intentionally used as a lightweight approximation rather than an exact geometric intersection test. While it does not capture fine-grained object geometry, it is sufficient for suppressing visually noticeable object interpenetration during scene layout optimization, while maintaining low computational cost.

Weighted objective (object-wise candidate scoring)

Importantly, the local search is performed independently for each object. For object i and a candidate placement p_i^k, three per-candidate terms $f_1(i, p_i^k)$, $f_2(i, p_i^k)$, and $f_3(i, p_i^k)$ are defined, corresponding to design fidelity, stability, and non-overlap, respectively. The combined score for candidate selection is computed as

$$F(i, p_i^k) = \alpha_1 f_1(i, p_i^k) + \alpha_2 f_2(i, p_i^k) + \alpha_3 f_3(i, p_i^k),$$

with default weights $\alpha_1 = 1.0$, $\alpha_2 = 2.0$, and $\alpha_3 = 1.5$. This configuration slightly prioritizes placement plausibility (stability and non-overlap) while preserving the original 2D design intent. In practice, these values produced stable results across common indoor furniture surfaces (e.g., floor, table, and bed) without requiring per-scene parameter tuning.

3.5 Trigger–Action Game Logic Editing

3.5.1. Trigger–Action Model

In addition to geometric layout design, game logic editing is a crucial component of AR game stage creation. To support the efficient construction of basic interactive behaviors, this work adopts a Trigger–Action logic model, which represents gameplay interactions as mappings from spatial triggers to corresponding actions.

The Trigger–Action model is widely used in commercial game engines such as Unity and Unreal, and has also been applied to game stage creation support systems in prior work by Ng et al.[3], demonstrating its effectiveness in terms of intuitiveness and usability for interaction design.

In the proposed system, Triggers are implemented using collision volumes in 3D space. When the player (i.e., the mobile device) enters or exits such a volume, the corresponding trigger event is detected. To approximate the player's physical presence in an AR environment, the system models the player using an axis-aligned bounding box centered on the device, with dimensions of 0.5 × 0.5 × 2.0 m, which is used for trigger detection.

In addition to spatial triggers, this work also implements an On Touch trigger to simulate explicit interaction behaviors such as button pressing. As illustrated in Figure 3.11, the system estimates the root position of the user's hand in the camera view using hand pose detection provided by MediaPipe [22]. A ray is cast from the camera position toward the detected hand position, and an intersection between the ray and the collision volume of a trigger object is regarded as the activation of the On Touch trigger.

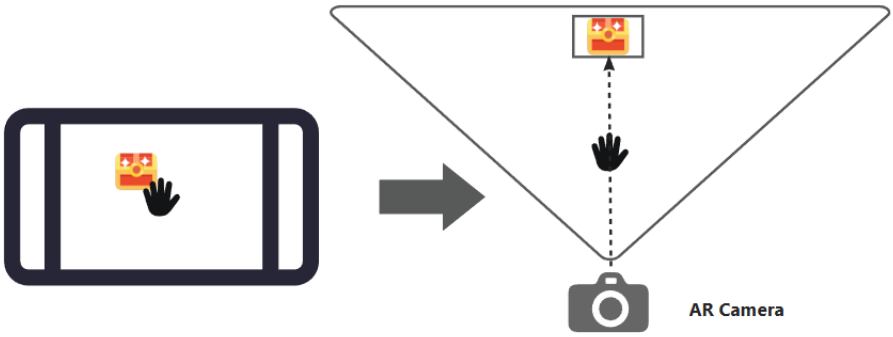


Figure 3.11 Principle of On Touch Trigger Implementation.

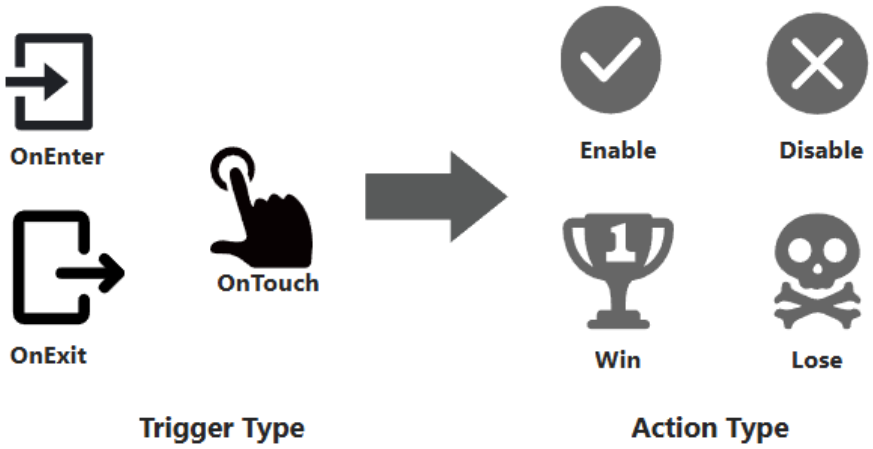


Figure 3.12 The types of Trigger-Action.

In the proposed system, three basic triggers and four action types are implemented, as illustrated in Figure 3.12. The supported trigger types are:

- On Enter: triggered when the player enters the trigger volume;
- On Exit: triggered when the player exits the trigger volume.
- On Touch: triggered when the player's hand enters the trigger volume.

An Action specifies the system response executed upon trigger activation. The supported action types in this work include:

- Win: mark the game as completed successfully;
- Lose: mark the game as failed;
- Enable: activate (make an object visible and enable its game logic) a target object;
- Disable: deactivate (make an object invisible and disable its game logic) a target object.

By combining triggers and actions, designers can construct basic yet functional gameplay flows. For example:

- Bomb object: On Enter → Lose;
- Key object: On Enter → Enable (Treasure Chest),
  Treasure Chest object: On Enter → Win.

This logic model provides a simple and intuitive abstraction, allowing common AR gameplay interactions to be authored without requiring complex scripting, while maintaining sufficient expressive power for stage design.

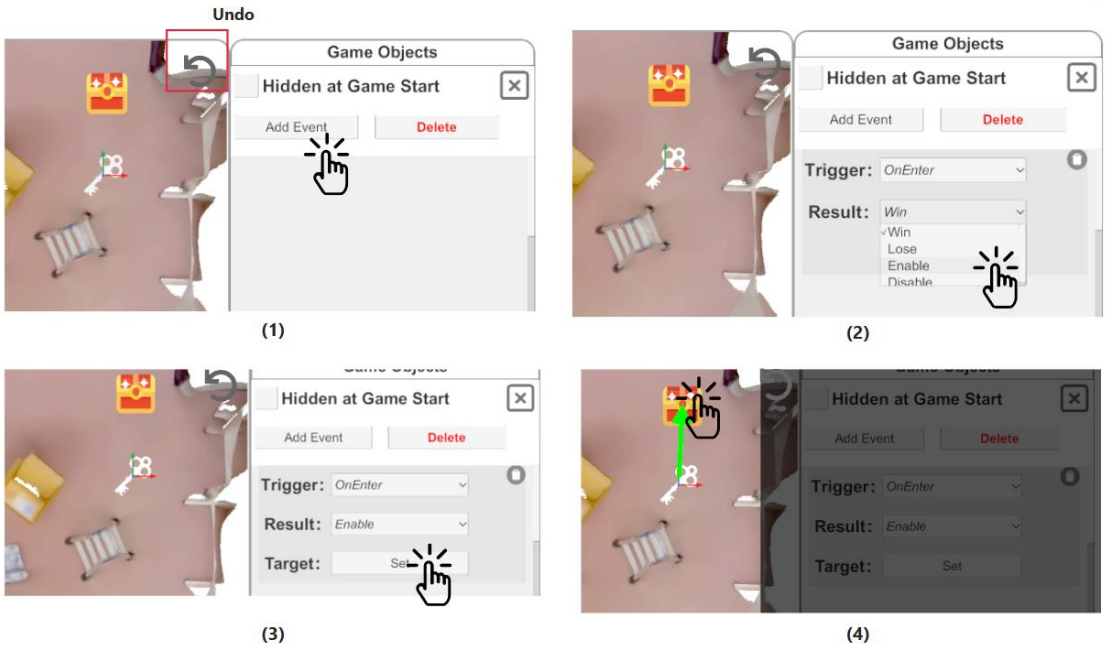


Figure 3.13 An example of creating a trigger–action.

This configuration realizes a basic gameplay logic in which the treasure chest is activated and becomes visible when the player interacts with the key and the trigger condition is satisfied.

To construct a Trigger–Action interaction, the designer first assigns an event to the object intended to serve as a trigger, as shown in Figure 3.13(1).

Next, the trigger condition and the corresponding action are specified by selecting OnEnter as the trigger type and Enable as the action type, as illustrated in Figure 3.13(2).

The user then clicks the Set button next to the Target field to enter the target selection mode, as shown in Figure 3.13(3).

Finally, the user selects the treasure chest object and assigns it as the target of the Enable action, thereby completing the action configuration, as shown in Figure 3.13(4). The green lines in the figure provide a visual representation of the event logic in which the key serves as the trigger, the action is Enable, and the target is the treasure chest. This logic visualization will be described in detail in the following subsection.

In addition, the system supports an undo function. By clicking the button located at the upper-right corner of the Top-Down View, the user can revert the most recent operation. Supported undo operations include adding, deleting, and moving objects, as well as adding or removing events and modifying object properties.

3.5.2. Logic Visualization

To help users understand the authored game logic, the system provides visual cues that explicitly represent Trigger–Action relationships. For example, green arrows are used to indicate Enable relationships, while red arrows represent Disable relationships. Objects associated with Win or Lose actions display corresponding icons above them to indicate game completion or failure conditions.

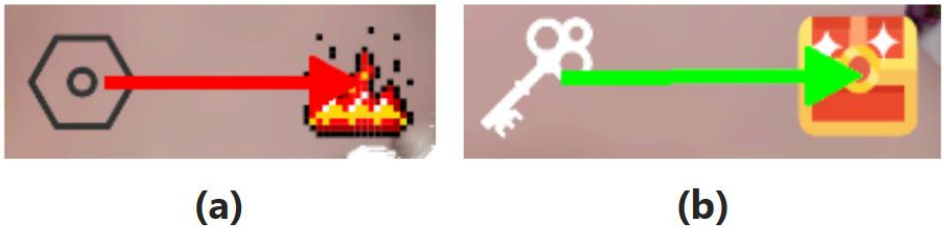


Figure 3.14 The logic visualization. (a)Red arrows indicate Disable relationships, (b)Green arrows indicate Enable relationships.

3.5.3. Consistency Between 2D and AR

The Trigger–Action logic can be authored in both the 2D design and the AR in-situ editing. Logic defined during the 2D stage is stored together with the spatial properties of objects as part of the stage data. When the 2D layout is mapped into the 3D environment and the system transitions to AR mode, the logic information is fully preserved and automatically applied to the corresponding objects in the AR space.

This design allows users to directly verify the logic defined in the 2D stage during AR gameplay. In addition, users can further refine the logic in the AR stage by modifying Action bindings, thereby supporting iterative design and validation across editing steps.

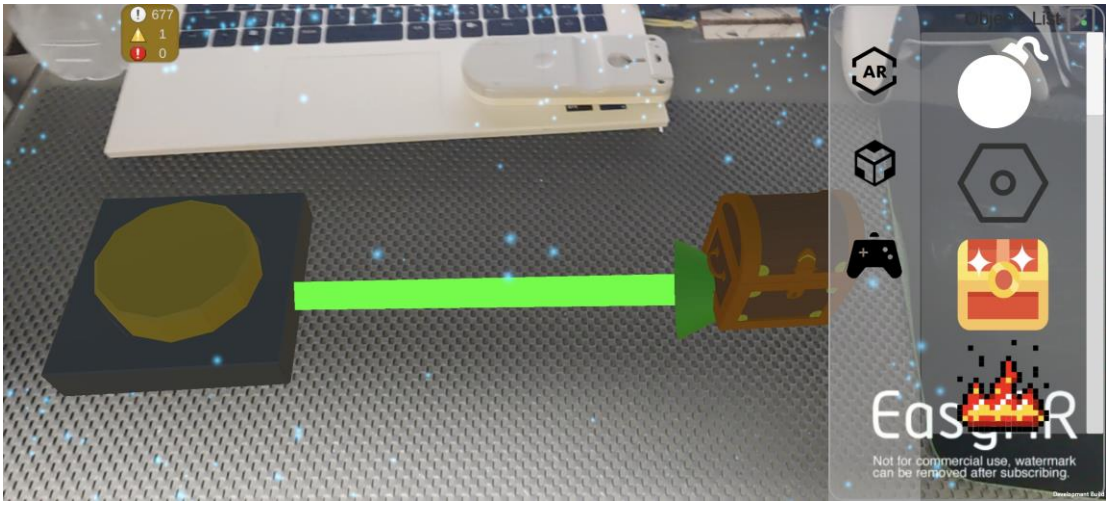


Figure 3.15 The logic visualization in AR.

As shown in Figure 3.15, the left button is configured as a trigger. When the action type is set to Enable and the treasure chest on the right is assigned as the action target, the editor visualizes the Trigger–Action relationship using a green arrow.

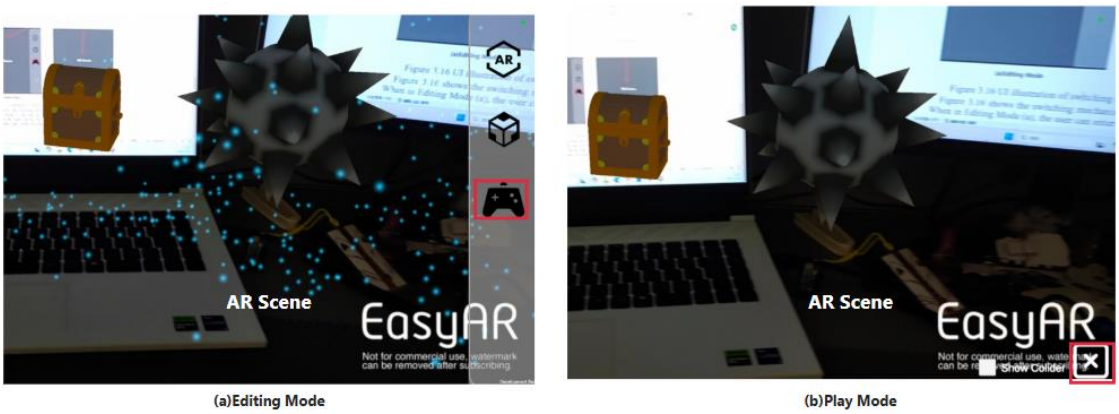
3.6 Play Mode

The system provides a Play mode for immediate gameplay validation. In Play mode:

- the editing user interface is hidden to avoid visual occlusion;

- Trigger detection is activated using a $0.5 \times 0.5 \times 2.0$ m player volume to approximate the player's physical presence;
- users can seamlessly switch from editing mode to Play mode (As shown in Figure 3.16) and quickly return to editing mode to continue iterative refinement.

This seamless mode switching enables designers to repeatedly test and adjust stage layout and game logic without interrupting the creation workflow.



(a)Editing Mode (b)Play Mode

Figure 3.16 UI illustration of switching between Editing Mode and Play Mode.

Figure 3.16 shows the switching mechanism between Editing Mode and Play Mode. When in Editing Mode (a), the user can enter gameplay by clicking the gamepad icon indicated by the red box on the right. During Play Mode (b), clicking the button indicated by the red box in the lower-right corner switches the system back to editing mode.

The blue particles shown in Figure 3.16(a) represent AR anchors used for spatial localization in the AR environment. These anchors are displayed for debugging purposes and can be hidden from the user's view during normal operation.

3.7 In-situ AR design

In the Edit Mode of the AR Editor, users can design directly in a first-person view by creating and manipulating virtual objects through gestures, enabling in-situ AR design. The overall workflow is illustrated in Figure 3.17. In Edit Mode, the user can tap the cube icon located on the right side of the screen (Figure 3.17(a)) to open the virtual object list. A virtual object can then be selected from the list and placed into the AR space via a drag-and-drop interaction, as shown in Figure 3.17(b).

By tapping an existing object in the scene, the user can select the object (Figure 3.17(c)). Once selected, the object's size and position can be adjusted using gesture-based interactions. The supported basic gestures are illustrated in Figure 3.17(d): a two-finger pinch gesture is used to scale the object, a two-finger horizontal movement rotates the object, a single-finger drag translates the object, and a double tap with a single finger performs an undo operation.

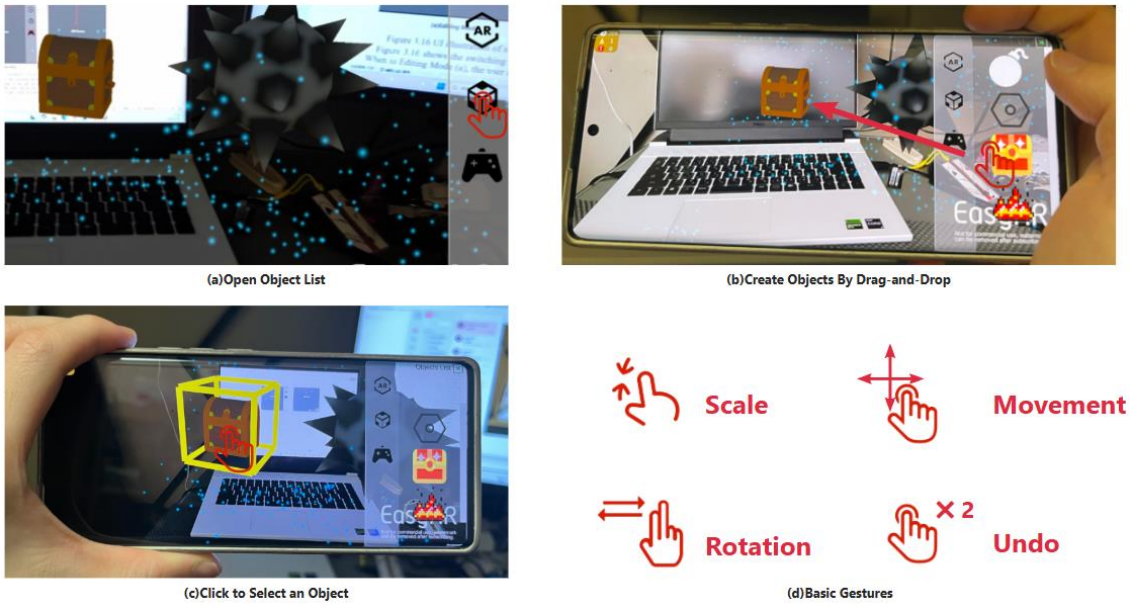


Figure 3.17 In-situ Design Workflow.

Chapter 4
Experimentation

This chapter explains the design of the comparative user study conducted to evaluate the proposed workflow. It describes the experimental purpose, participant grouping, tasks and procedures, device and implementation settings, and the data collection methods used for subsequent analysis.

4.1 Purpose

The purpose of this experiment is to evaluate the effectiveness of the proposed AR game stage creation workflow that integrates 2D Layout Design, Mapping, and In-situ Design in practical use. The experiment focuses on the following research questions:

- Whether the proposed workflow can reduce the cognitive load involved in AR game stage creation;
- Whether it improves the efficiency of overall layout planning;
- Whether it helps users more clearly express their design intentions and complete playable AR game stages;
- Whether it shows advantageous trends in terms of production efficiency and user experience compared to conventional workflows.

To this end, a controlled comparative experiment was conducted, comparing user behaviors and subjective evaluations under two different stage creation workflows: the proposed “2D Layout Design + Mapping + In-situ Design” workflow, and a baseline workflow based on paper prototyping followed by in-situ AR design, as represented by the work of Ng et al.[3]

4.2 Participants

A total of 11 participants were recruited for the experiment, including university students and practitioners with some experience in game development or design. All participants had basic experience using smartphones but had no prior systematic exposure to AR game level editing tools. Among them, one participant had approximately three years of professional experience in the game industry with relevant level design background.

Participants were divided into three groups according to their roles in the experiment:

Group A (2D + AR group): 4 participants

Participants used the proposed 2D top-down layout editor together with the in-situ AR editing tool to create game levels.

Group B (Paper + AR group): 4 participants

Participants first designed level layouts using paper prototypes and then directly created the levels in AR using in-situ editing.

Group C (Playtesting evaluation group): 3 participants

Participants in this group did not engage in level creation. Instead, they evaluated the completed levels from a player and design assessment perspective through gameplay and scoring.

Group C included one professional level designer with approximately three years of experience, one creator with game development experience, and one undergraduate student majoring in game design–related fields. This group was used to provide third-party evaluations of level fun, playability, and goal clarity.

Participant demographics such as age range, gender distribution, and gaming frequency were not treated as primary comparison factors and were therefore not further differentiated. All participants received an explanation of the experiment and provided informed consent prior to participation.

4.3 Experimental Conditions and System Implementation

4.3.1. Experimental Groups and Procedures

The experiment compared the following two AR game stage creation workflows:

Group A (2D + AR)

Participants first used the proposed 2D top-down editor to design the overall stage layout and Trigger–Action game logic. In this editor, a prepared top-down floor plan of the experimental room was provided as the background, which was identical for all participants and corresponded to the real environment used in the experiment. Participants did not capture the layout themselves.

After completing the 2D design, the layout was mapped into the AR space, followed by fine-grained adjustments using in-situ AR editing. Finally, participants entered Play Mode to validate the stage through gameplay.

Group B (Paper + AR)

Participants first conceptualized the stage layout and game logic using paper prototyping. To ensure comparable initial conditions with Group A, participants were provided with an electronic drawing device (iPad) displaying the same top-down floor plan of the experimental room as used in Group A. The drawing was performed using the Procreate[23] application with default brush settings.

Participants manually sketched the stage layout on this diagram while observing the real environment. Basic operations of the drawing tool could be learned within approximately one minute, and this familiarization time was not included in the subsequent experimental measurements.

Afterward, participants proceeded directly to stage creation and gameplay validation using in-situ AR editing, without an intermediate digital 2D layout editing stage.

Both workflows shared identical criteria for stage completion, available editing functions, and gameplay validation procedures. The primary difference between them was whether a dedicated digital 2D layout editing stage was introduced prior to AR-based editing.

Group C (Playtesting evaluation)

Participants in Group C did not participate in stage creation. Instead, they played and evaluated the completed stages produced by Groups A and B from the perspectives of player experience and stage design, including fun, goal clarity, and overall playability.

To reduce potential evaluation bias, stages were presented in an anonymized manner such that participants were not informed whether a stage was created using the proposed

workflow (Group A) or the baseline workflow (Group B). The presentation order of stages was randomized for each evaluator.

Each participant in Group C played and evaluated all eight stages created by participants in Groups A and B, ensuring that every stage received an equal number of third-party evaluations. Ratings were collected using a unified evaluation form with consistent criteria across all stages.

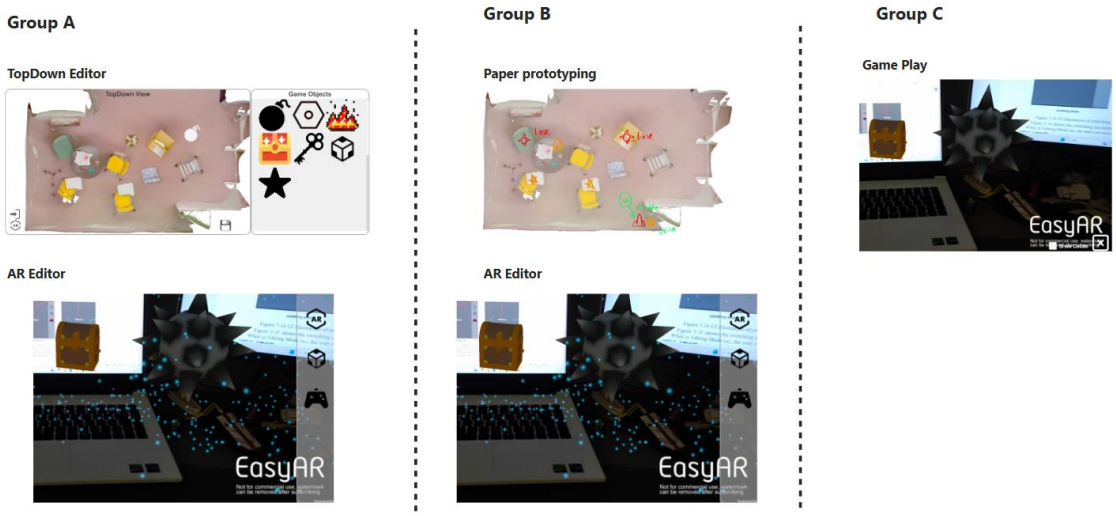


Figure 4.1 Initial conditions for the three experimental groups.

Overall, the initial conditions of the three groups are illustrated in Figure 4.1. Group A designed the stage layout using a prepared top-down floor plan and the proposed 2D editor, and then refined the stage in the AR editor. Group B manually sketched the stage layout based on the same prepared top-down floor plan. Group C only participated in playtesting of the AR stages.

4.3.2. Experimental Device

All experiments were conducted using the same smartphone model, Honor 60(Snapdragon 778G), to run the experimental system for 2D editing, in-situ AR editing, and gameplay. The device supports plane detection and spatial tracking via AR Foundation and was able to run all required system functions stably.

For Group B, an iPad Pro (11-inch, 2nd generation) was additionally provided as an electronic drawing device for paper prototyping and manual layout sketching.

4.3.3. System Implementation and Logging

The experimental system was implemented using the Unity engine and the AR Foundation framework. Its main functional modules include:

- Loading and alignment of reconstructed environment meshes;
- A 2D top-down stage editing interface;
- Automatic mapping from 2D layouts to AR space;
- In-situ AR editing and Trigger–Action game logic editing;
- Game Play mode;

- Automatic operation logging module.

During the experiment, the system automatically recorded key user actions, including object placement, movement, deletion, logic editing, and mode switching. After completing the experiment, participants filled out subjective questionnaires to evaluate their creation experience and the resulting levels.

Operation logs were exported in a structured format, as shown in Figure 4.2, containing information such as event type, timestamp, target object, and current mode. These logs were used for subsequent behavioral analysis.

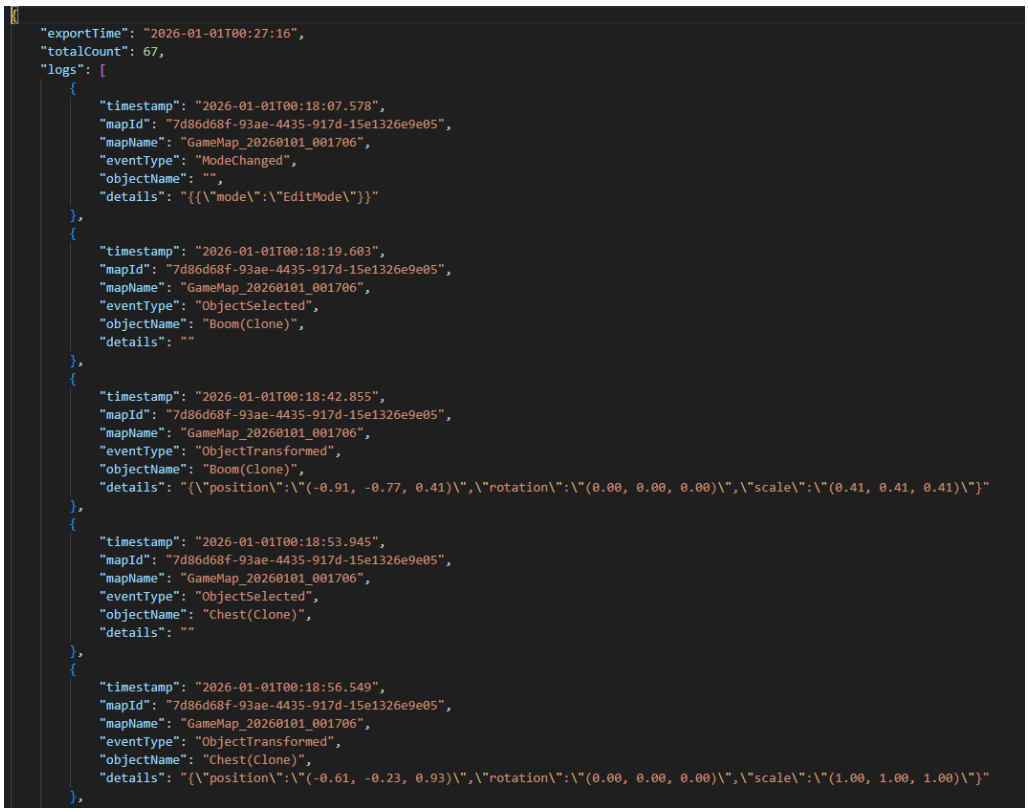


Figure 4.2 Example of an operation log.

4.4 Experimental Tasks

Participants in Groups A and B were required to create a playable AR game stage within a given real indoor environment. The task requirements were as follows:

- The stage must include at least one win condition and one lose condition;
- The stage must contain at least one Trigger–Action game logic;
- The completed stage must be validated through actual gameplay.

The unified experimental procedure was as follows:

1. Explanation of the experiment and system operation tutorial (including a 5-minute practice session);
2. Stage conceptualization and creation according to assigned group workflow;
3. Gameplay validation in the AR environment;
4. Completion of questionnaires and provision of free-form feedback.

The experimental environment was a standard indoor room containing common

furniture such as tables and chairs, with clearly defined walkable areas. The room size and layout were kept consistent across all participants.

The experimental procedure for Group C consisted only of gameplay and evaluation.

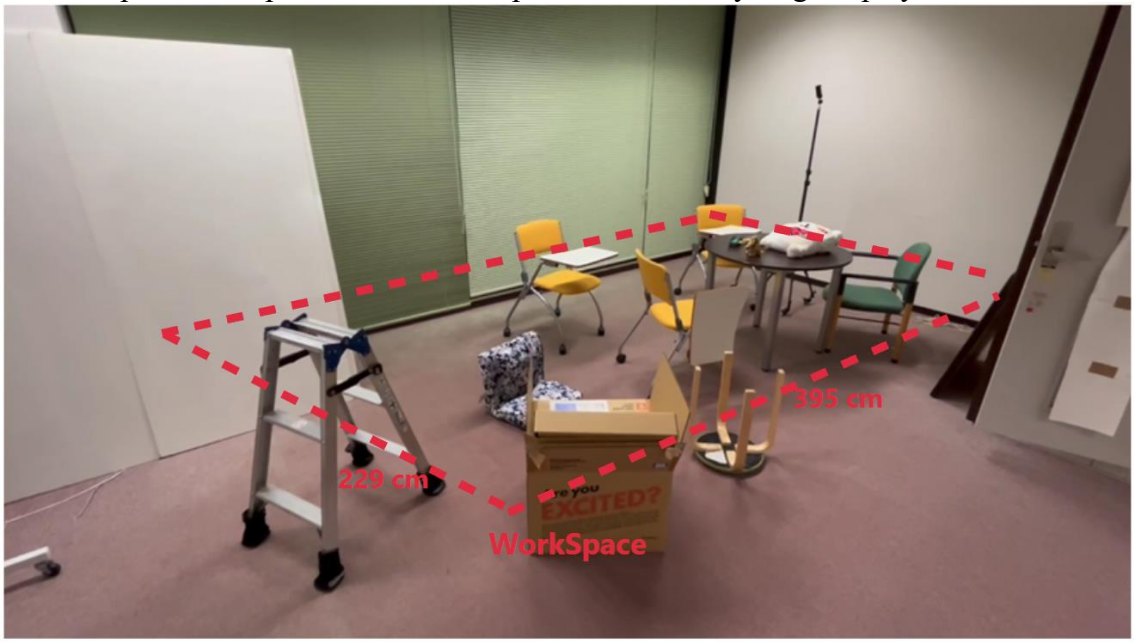


Figure 4.3 The experimental environment.

4.5 Data Collection Methods

Two types of data were collected during the experiment:

(1) Operation Log Data (Quantitative)

The system recorded user operations during stage creation, including object placement, movement, deletion, logic editing, and mode switching. In the subsequent analysis, the number of object movement and deletion operations during the AR stage was used as a proxy measure for trial-and-error cost, reflecting efficiency differences between workflows in the spatial adjustment phase.

(2) Subjective Questionnaire and Rating Data (Qualitative + Quantitative)

Subjective evaluation data were collected via questionnaires, covering the following dimensions:

- Comprehensibility and coherence of the creation workflow;
- Perceived operational burden and trial-and-error effort;
- Ability to express design intentions and control spatial layout;
- Perceived quality, playability, and immersion of the final stage.

The questionnaire design was informed by established evaluation frameworks such as the SUS, NASA-TLX, and commonly used dimensions in creative support system research, and was adapted to the characteristics of AR game stage creation tasks. A 5-point Likert scale was used.

In addition, participants in Group C used an independent rating scale to evaluate stage fun, playability, and goal clarity from player and design assessment perspectives. These results were used as a reference for comparison with the self-evaluations provided by creators.

4.6 Experimental Results

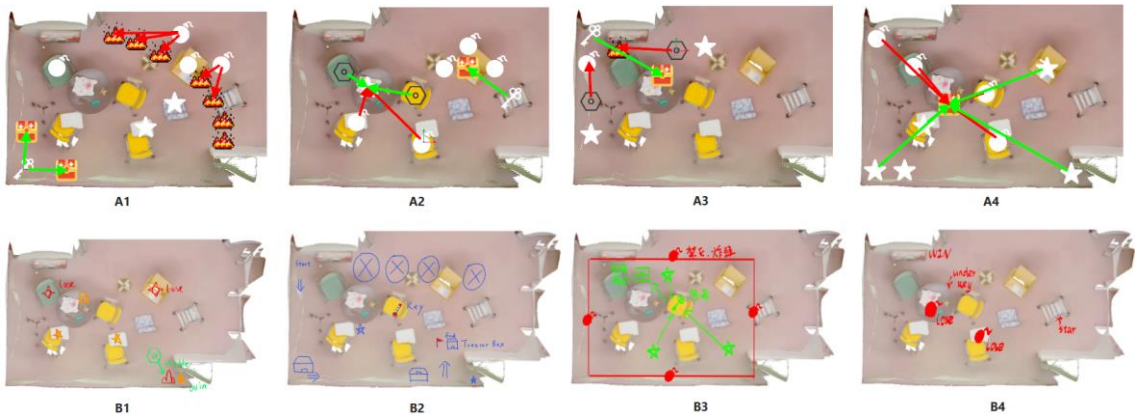


Figure 4.4 All experimental results related to stage layout.

Figure 4.4 shows all experimental results from the layout design stage for Group A and Group B.

In Group A, layouts were created using a predefined top-down floor plan together with a set of predefined icons.

In Group B, layouts were drawn on the same predefined top-down floor plan but using freehand sketching without restrictions on representation style.

Chapter 5
Evaluation

This chapter reports and analyzes the experimental results using operation logs, subjective questionnaires, and third-party playtesting ratings. It compares the proposed workflow with the baseline in terms of production efficiency, trial-and-error behavior, perceived user experience, and stage outcome quality.

5.1 Evaluation Metrics

This chapter evaluates the proposed AR game stage creation workflow based on the operation log data and subjective questionnaire results collected during the experiment. The evaluation focuses on the differences observed between workflows in terms of production efficiency, operation behavior characteristics, subjective experience, and the quality of the final game levels. The analysis combines descriptive statistics with behavioral evidence.

The evaluation is conducted from the following four perspectives:

Production Efficiency

This metric examines the time required for overall stage layout planning, as well as the total production time from the start of editing to the completion of gameplay validation. Operation counts are also considered to analyze editing efficiency within a given time period.

Operational Load and Trial-and-Error Cost

Operational load is analyzed by counting behaviors such as object movements and deletions recorded in the operation logs, and by examining questionnaire responses related to perceived cognitive load, frustration, and the need for trial-and-error.

Design Expression and Controllability

This aspect evaluates whether participants were able to effectively express their stage design intentions, control spatial relationships between objects, and conveniently modify or adjust the design when issues were identified.

Stage Quality and Gameplay Experience

The quality of the resulting stages is assessed from both creators' self-evaluations and third-party playtesting evaluations, focusing on overall stage quality, playability, and immersion.

Stage quality is treated as a holistic indicator that reflects players' first impressions of a stage. It captures the overall completeness and presentation quality of the stage, including factors such as content richness, spatial organization, and the visual coherence of object placement.

Playability evaluates the extent to which the stage functions effectively as a game and provides an enjoyable gameplay experience for players. It focuses on whether the stage supports clear goals, meaningful interactions, and smooth progression from a gameplay perspective.

Immersion represents the distinctive value of an AR game stage, assessing how naturally virtual content is integrated with the real physical environment. It reflects

whether the alignment between virtual elements and real space supports a convincing and seamless AR experience.

These three evaluation dimensions are hierarchically related. Stage quality assesses whether the stage is competent as a piece of virtual interactive content, playability evaluates its fundamental value as a game, and immersion further examines its unique value as an AR experience that blends virtual and physical spaces.

All subjective evaluations were measured using a 5-point Likert scale.

5.2 Production Efficiency Analysis

Participant	*Total Production Time (min)*	*Macro-Stage Planning Time (min / %)*	*Total Operation Count*
A1	21.5	2.5 / 11.6%	265
A2	30.2	5.0 / 16.6%	457
A3	26.8	4.5 / 16.8%	338
A4	28.3	4.0 / 14.1%	290
Group A(Mean)	**26.7**	**4.0 / 15.0%**	**338**
B1	29.8	4.0 / 13.4%	207
B2	41.0	9.5 / 23.2%	328
B3	36.2	7.5 / 20.7%	245
B4	34.1	10.0 / 29.3%	271
Group B(Mean)	**35.3**	**7.8 / 22.1%**	**263**

Table 5.1 Production Time and Operation Statistics per Participant.

5.2.1. Total Production Time

Differences in efficiency were also observed in the total production time from the start of editing to the completion of gameplay validation:

- Group A: Mean time of 26.7 minutes
- Group B: Mean time of 35.3 minutes

Although no strict time limits were imposed during the experiment, these results suggest that front-loading macro-level planning into the 2D layout stage contributes to shortening the overall production workflow.

5.2.2. Macro-Stage Planning Time

During the step of overall stage layout planning, a clear difference in required time was observed between the two groups.

- Group A: Mean time of 4 minutes
- Group B: Mean time of 7.8 minutes

These results indicate that introducing a 2D top-down layout editor enables participants to complete macro-stage structural planning more efficiently. In contrast, paper prototyping requires additional confirmation and adjustment when transitioning to the AR space, which prolongs the effective time spent in the planning phase.

5.2.3. Operation Count and Operation Density

Analysis of the operation logs revealed clear differences in the total number of editing operations performed by each group. Group A performed a higher mean number of operations (338) than Group B (263). When considered together with total production time, participants in Group A completed more editing operations within a shorter time span, resulting in a higher operation density.

This higher operation density indicates a more continuous and uninterrupted editing process. Because macro-stage layout planning was largely completed during the 2D layout design stage in Group A, subsequent operations in the AR phase were more goal-directed and focused primarily on fine-grained spatial alignment and experiential validation.

In contrast, participants in Group B were required to perform both macro-stage planning and spatial adjustment directly within the first-person AR view. As a result, their editing process was more fragmented, with frequent interruptions caused by repeated confirmation, repositioning, and rollback actions. This difference in operational patterns suggests that separating macro-stage planning from in-situ AR editing contributes to more efficient and coherent editing behavior during the AR phase.

5.3 Operation Behavior and Trial-and-Error Cost Analysis

5.3.1. Log-Based Behavior Quantification

The system automatically records interaction logs during both editing and playtesting, including events such as object placement, movement, deletion, and switching between edit/play modes. In this study, we treat these logs as behavioral traces of the design process, enabling analysis of differences in trial-and-error behavior under different workflows.

In particular, we use object movement and deletion operations during the AR phase as proxy measures of trial-and-error cost, since these actions often indicate repeated adjustments caused by insufficient spatial understanding or unclear layout planning.

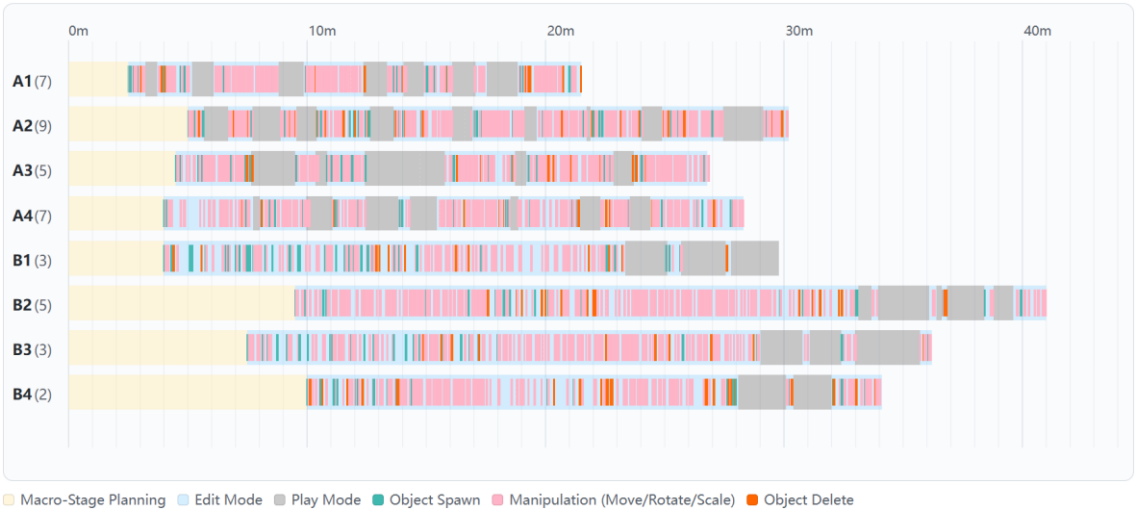


Figure 5.1 Stacked bar chart comparing the distribution of operation types between Group A and Group B. The numbers shown next to each group indicate the number of Play Mode sessions.

5.3.2. Comparison of Trial-and-Error Behavior in AR

The log analysis indicates that Group A exhibited fewer iterative adjustments during the AR phase than Group B:

- Group A: Manipulation(movement/rotation/scale) accounted for approximately 31.9%, and deletion operations accounted for approximately 4.7%.
- Group B: Manipulation(movement/rotation/scale) accounted for approximately 38.5%, and deletion operations accounted for approximately 6.8%.

In terms of behavioral interpretation, Group A completed most of their layout planning in the 2D top-down view before entering the AR phase; therefore, the AR phase was mainly used for fine alignment and experiential validation. In contrast, Group B had to perform both macro-stage planning and fine-grained adjustments directly in the AR first-person view, leading to more frequent positional trial-and-error and object re-placement.

5.3.3. Comparison of Playtest Willingness

As shown in Figure 5.1, after completing macro-stage planning, participants in Group A tended to enter Play Mode earlier and more frequently. Specifically, the first entry into Play Mode for all Group A participants occurred within the first 10 minutes, whereas for Group B it occurred after 20 minutes. Moreover, Group A conducted more playtest sessions overall (7 sessions on average) compared with Group B (3 sessions on average).

These results provide evidence for the effectiveness of Macro-Stage Planning and Mapping, which help participants quickly establish the core stage flow and allocate more time to playtesting and iterative refinement.

5.4 Subjective Questionnaire Results (Creator Self-Report)

To examine differences in creators' subjective experiences between the two workflows, we analyzed questionnaire responses along three dimensions: reuse intention, frustration, and need for trial-and-error. The results are summarized in Table 5.2.

Reuse intention measures participants' willingness to use the same workflow again in future stage-creation tasks, reflecting its overall acceptance and perceived usefulness.

Frustration captures negative emotional responses arising from difficulties such as repeated adjustments, operation breakdowns, or workflow inefficiencies during the creation process.

Need for trial-and-error reflects the extent to which participants felt it was necessary to rely on repeated experimentation and modification in order to complete the stage design.

Questionnaire Item	*Group A (Mean)*	*Group B (Mean)*
Reuse Intention	**4.75**	3.75
Frustration	**1.5**	2.5
Need for Trial-and-Error	**2.0**	2.75

Table 5.2 Subjective Questionnaire Results (Creator Self-Report).

As shown in Table 5.2, Group A reported a higher reuse intention, whereas Group B reported higher levels of frustration and need for trial-and-error. These subjective results are consistent with the differences in AR-phase trial-and-error behavior observed in the operation log analysis.

This version is fully consistent with academic writing conventions and pairs cleanly with your Section 5.3 results.

5.5 Stage Outcome Quality and Experience Evaluation

To evaluate the impact of different creation workflows on the quality and experiential performance of the final stages, we conducted a comparative analysis from two perspectives: creator self-evaluation and third-party playtest evaluation. The results are summarized in Table 5.3.

Evaluation Source	*Evaluation Item*	*Group A (Mean)*	*Group B (Mean)*
Creator self-report	Overall rating	**4.0**	3.25
Creator self-report	Playability	4.5	4.5
Creator self-report	Immersion	**4.0**	3.5
Third-party playtest (Group C)	Overall rating	**3.75**	3.25
Third-party playtest (Group C)	Playability	**4.25**	3.5
Third-party playtest (Group C)	Immersion	**3.5**	3.0

Table 5.3 Comparison of Stage Outcome Quality and Experience Ratings.

Metric Definitions

In this evaluation, overall rating represents a holistic judgment of stage quality, taking into account structural completeness, experiential coherence, and perceived overall polish.

Playability reflects the clarity of gameplay mechanics, the appropriateness of interaction feedback, and whether the stage supports sustained engagement during play.

Immersion describes the extent to which the stage's spatial presentation, pacing, and interaction consistency support players' sense of involvement and presence.

5.5.1. Creator Self-Evaluation Results

As shown in Table 5.3, Group A reported higher mean scores than Group B in both overall rating and immersion, while the two groups received identical scores for playability. Specifically, the mean overall rating for Group A was 4.0, compared with 3.25 for Group B, and the mean immersion score for Group A was 4.0, compared with 3.5 for Group B. In contrast, both groups reported the same playability score of 4.5.

These results indicate that, from the creators' perspective, differences between workflows were more strongly reflected in the overall quality and experiential aspects of the completed stages, rather than in the perceived playability of the core gameplay mechanics. This suggests that both workflows were sufficient to support the creation of functionally playable stages because the underlying gameplay logic and interaction mechanisms were identical across conditions.

In contrast, differences in the creation process mainly affected aspects related to spatial organization and layout clarity, which contributed to differences in overall quality and immersion.

5.5.2. Third-Party Playtest Evaluation Results (Group C)

In the third-party playtest evaluation, participants in Group C assessed the completed stages from the perspectives of player experience and level design. As shown in Table 5.3, Group A received higher mean scores than Group B across all three evaluation items. The mean overall rating was 3.75 for Group A and 3.25 for Group B. For playability, Group A scored 4.25, compared with 3.5 for Group B, and for immersion, Group A scored 3.5, compared with 3.0 for Group B.

Compared with creator self-evaluations, third-party ratings were generally lower, reflecting a more conservative assessment tendency among external evaluators. Nevertheless, the relative trend between the two groups remained consistent, indicating that the effects of workflow differences on stage quality and player experience were not only perceived by the creators themselves but were also observable to external players.

5.6 Summary

By integrating the analyses of production efficiency, operation logs, subjective questionnaires, and third-party playtesting evaluations, the results consistently indicate that the proposed "2D layout design + mapping + in-situ AR editing" workflow outperforms the baseline workflow across multiple dimensions.

From the perspective of production efficiency, Group A completed stage creation in a shorter overall time while performing a larger number of effective editing operations, resulting in higher operation density and a more continuous design process. This suggests that introducing an explicit 2D macro-level layout planning stage before entering the cognitively demanding AR editing phase helps designers more efficiently understand and

construct the overall stage structure.

Operation log analysis further shows that the proposed workflow significantly reduces trial-and-error behavior during the AR phase. Participants in Group A exhibited fewer repetitive adjustment and deletion operations during in-situ AR editing, and entered Play Mode earlier and more frequently for gameplay validation. These behavioral patterns indicate that separating macro-level layout planning from fine-grained AR adjustments allows designers to establish the core stage structure earlier and allocate more effort to validation and refinement rather than error correction.

The subjective questionnaire results are consistent with the behavioral analysis. Participants using the proposed workflow reported higher reuse intention and lower levels of frustration and trial-and-error burden, suggesting that the workflow not only improves objective efficiency but also reduces cognitive and emotional costs during the stage creation process.

In terms of final stage quality and experiential evaluation, Group A achieved higher scores in both overall quality and immersion in both creator self-evaluations and third-party playtesting, even though both workflows were capable of supporting the creation of functionally playable AR game stages. This indicates that workflow design does not substantially affect the playability of core gameplay mechanics (under controlled interaction logic), but instead influences higher-level outcomes such as overall quality and immersion by shaping spatial organization and layout clarity.

Overall, these results support the central claim of this work: introducing a low-cognitive-cost macro-level stage representation prior to in-situ AR editing can effectively improve the efficiency, stability, and experiential quality of AR game stage creation, without sacrificing creative freedom or gameplay functionality.

Chapter 6
Conclusion

This chapter summarizes the main findings and contributions of the study and discusses its limitations. It also outlines future directions for improving robustness, mapping optimization, logic expressiveness, and intelligent authoring support.

6.1 Summary of the Research

This study addresses the high trial-and-error cost and cognitive burden of creating AR game stages that must align with real indoor environments. We proposed an AR stage creation support method based on real room layouts, introducing a multi-stage workflow that integrates 2D top-down layout design with in-situ AR editing. By allowing designers to externalize macro-stage structure in a low-cost 2D representation, and reserving first-person AR interaction for necessary spatial calibration and experience validation, the proposed workflow aims to balance global planning efficiency with fine-grained adjustment under real-world constraints.

From an implementation perspective, the system was built with Unity and AR Foundation, and uses a mesh representation derived from 2DGS reconstruction as geometric constraints. For the mapping stage, we adopted an object-wise strategy based on "projection + local search ," which attempts to preserve the designer's 2D intent while reducing common placement failures such as floating objects, penetration, and unreasonable overlaps. For logic authoring, we employed a low-barrier Trigger–Action model and provided visualization support and seamless switching between edit and play modes to facilitate rapid iteration.

Comparative user study results indicate consistently positive trends for the proposed workflow relative to the baseline workflow (paper prototyping + in-situ AR editing). In terms of production efficiency, participants were able to complete stage creation and validation in a shorter overall time. In terms of operation behavior, the proportion of trial-and-error actions during the AR phase (e.g., object manipulation and deletion) was lower. Subjective questionnaire results showed higher reuse intention and lower frustration and perceived need for trial-and-error. Finally, both creator self-evaluations and third-party playtest evaluations showed a favorable trend for stages produced with the proposed workflow (while third-party ratings were generally more conservative). Taken together, these results support the claim that front-loading macro planning into a 2D layout stage and bridging it to AR via automatic mapping can improve the efficiency and usability of mobile AR stage creation in practice.

6.2 Limitations

Despite the promising trends observed in the evaluation, this study has several limitations that should be addressed in future work.

First, reconstruction accuracy and spatial alignment errors can affect mapping stability and placement precision. In mobile AR settings, tracking drift, reconstruction noise, and incomplete mesh details may lead to unstable support estimation, minor penetration artifacts, or position offsets, which in turn require compensatory manual adjustments during the AR phase.

Second, the proposed mapping strategy relies on object-wise local optimization. While this design is computationally efficient and practical for real-time use, it may struggle to ensure global consistency in dense layouts or in stages where object relations impose strong global constraints (e.g., multi-object mechanisms that jointly define navigable paths). As a result, users may still need repeated in-situ adjustments in complex scenarios.

Third, in terms of logic authoring, the Trigger–Action model lowers the barrier for building interactive behaviors, but its expressive power is limited. It cannot easily represent complex conditional branching, multi-state transitions, or higher-level gameplay structures, which constrains the complexity of stages that can be authored within the current system.

Finally, the user study sample size was limited, and the analysis primarily relied on descriptive statistics and behavioral trend analysis, rather than statistical significance testing. Therefore, this work emphasizes directional advantages observed in practical use rather than making definitive claims about the magnitude of performance improvements.

6.3 Future Work

To address these limitations, future work will proceed along four directions: environment understanding, mapping optimization, logic expressiveness, and intelligent authoring support.

First, at the level of environment modeling and spatial understanding, we plan to incorporate more robust geometric and semantic cues, such as plane-type recognition, stable support-surface detection, semantic segmentation, and walkable-area analysis. These improvements are expected to strengthen the mapping stage's ability to reason about placeability, navigability, and occlusion, thereby improving robustness and spatial consistency.

Second, on the mapping algorithm side, we will extend object-wise local optimization toward a hybrid local–global optimization framework. By introducing inter-object constraints and global structural objectives while maintaining computational efficiency, the system can better handle dense layouts, strongly coupled mechanisms, and complex spatial relationships, reducing the need for manual fine-tuning after mapping.

Third, for logic editing, we will explore richer authoring mechanisms such as logic template libraries, composable conditions, and visual state machines to improve expressiveness and scalability. We will also enhance logic visualization and debugging support so users can more easily understand, validate, and iterate on interactive structures.

Finally, to further lower the creation barrier—especially for non-expert users—we will investigate intelligent assistance based on historical design data and large language models. Potential functions include placement suggestions, logic structure recommendations, auto-completion, and error detection, enabling more efficient creation of diverse and complex AR game stages.

Acknowledgements

I would like to express my sincere gratitude to everyone who supported and guided me throughout this research.

First of all, I would like to thank my supervisors, Prof. Kazunori Miyata and Prof. Haoran Xie, for their invaluable guidance and continuous support. I am also deeply grateful to the senior members, peers, and junior members of the Miyata Laboratory for their help and encouragement.

Prof. Miyata and Prof. Xie provided me with a great deal of advice, including direction on the research topic and ideas for implementation. They also introduced me to related studies and examples, which were extremely helpful references for this work. Furthermore, through progress reports and paper discussions in our seminars, I had many opportunities to read and learn from recent papers from venues such as SIGGRAPH and Eurographics, which broadened my perspective significantly.

I would also like to thank the senior students in the laboratory for their advice on both research and job hunting. In addition, I am grateful to my peers and junior students, with whom I shared my student life and spent many enjoyable and meaningful days.

Finally, I would like to express my heartfelt appreciation to everyone in the Miyata Laboratory who supported me in my daily life and research. Although my time here was only two years, it was a period during which I learned a great deal and grew as a researcher and as a person. Thank you very much.

Bibliography

[1] Niantic, Inc. *Pokémon GO*. Mobile Augmented Reality Game, released in 2016. Available at: https://pokemongo.com (accessed February 2026).

[2] Microsoft Corporation. Minecraft Earth. Mobile Augmented Reality Game, (2019). Available at: https://www.minecraft.net/en-us/article/minecraft-earth-coming-end (accessed February 2026).

[3] Ng, G., Shin, J. G., Plopski, A., Sandor, C., and Saakes, D. Situated Game Level Editing in Augmented Reality, Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction (TEI ’18), pp. 409–418 (2018).

[4] Liu, J., Zhang, R., Sawabe, T., Fujimoto, Y., Kanbara, M., and Kato, H. Integrating Spatial Design with Reality: An AR Game Scene Generation Approach, Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW 2024), pp. 715–716 (2024).

[5] Tahara, T., Seno, T., Narita, G., and Ishikawa, T. Retargetable AR: Context-aware Augmented Reality in Indoor Scenes based on 3D Scene Graph, Proceedings of the IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct 2020), pp. 249–255 (2020).

[6] Liu, H., Wang, Z., Mazumdar, A., and Mousas, C. Virtual Reality Game Level Layout Design for Real Environment Constraints, Graphics and Visual Computing, vol. 4, Article 200020 (2021).

[7] Sun, Q., Wei, L.-Y., and Kaufman, A. Mapping Virtual and Physical Reality, ACM Transactions on Graphics, vol. 35, no. 4, Article 64, (2016).

[8] Unity Technologies. Unity Real-Time Development Platform. Available at: https://unity.com (accessed February 2026).

[9] Unity Technologies. AR Foundation: Cross-platform Framework for Augmented Reality. Available at: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html (accessed February 2026).

[10] EasyAR Inc. EasyAR Sense: Augmented Reality SDK. Available at: https://www.easyar.com (accessed February 2026).

[11] Schönberger, J. L., and Frahm, J.-M. Structure-from-Motion Revisited, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), pp. 4104–4113, 2016.

[12] Furukawa, Y., and Ponce, J. Accurate, Dense, and Robust Multi-View Stereopsis, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol. 32, No. 8, pp. 1362–1376, 2010.

[13] COLMAP. COLMAP: Structure-from-Motion and Multi-View Stereo, Available at: https://colmap.github.io/ (accessed February 2026).

[14] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. KinectFusion: Real-Time Dense Surface Mapping and Tracking, Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2011), pp. 127–136, 2011.

[15] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Proceedings of the European Conference on Computer Vision (ECCV 2020), pp. 405–421 (2020).

[16] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering, ACM Transactions on Graphics (Proceedings of SIGGRAPH 2023), Vol. 42, No. 4, Article 139 (2023).

[17] Huang, B., Yu, Z., Chen, A., Geiger, A., and Gao, S. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields, SIGGRAPH 2024 Conference Papers, Association for Computing Machinery (ACM) (2024).

[18] Littlefair, G., Dutt, N. S., and Mitra, N. J. FlairGPT: Repurposing LLMs for Interior Designs, Computer Graphics Forum, vol. 44, no. 2 (Eurographics 2025) (2025).

[19] Aguina-Kang, R., Gumin, M., Han, D. H., Morris, S., Yoo, S. J., Ganeshan, A., Jones, R. K., Wei, Q. A., Fu, K., and Ritchie, D. Open-Universe Indoor Scene Generation using LLM Program Synthesis and Uncurated Object Databases, arXiv preprint arXiv:2403.09675 (2024).

[20] Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., and Wang, X. I-Design: Personalized LLM Interior Designer, in Computer Vision – ECCV 2024 Workshops, Springer, pp. 217–234 (2025).

[21] hbb1. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. GitHub repository. Available at: https://github.com/hbb1/2d-gaussian-splatting (accessed February 2026).

[22] Google. MediaPipe: Cross-platform Framework for Perception Pipelines. Available at: https://chuoling.github.io/mediapipe/ (accessed February 2026).

[23] Procreate. Procreate: digital painting app for iPad. Available at: https://procreate.com/ (accessed February 2026).