M.S. THESIS

Bubbleu: Empirical Design Study of Object Detection as an Augmented Reality Game Interaction Method

Bubbleu: 증강 현실 게임의 상호작용 방식으로서의 객체 인식의 실증적 설계 연구

2022 년 8 월

DEPARTMENT OF ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

김민지

Bubbleu: Empirical Design Study of Object Detection as an Augmented Reality Game Interaction Method

Bubbleu: 증강 현실 게임의 상호작용 방식으로서의 객체 인식의 실증적 설계 연구

지도교수 이 영 기

이 논문을 공학석사 학위논문으로 제출함

2022 년 7 월

서울대학교 대학원

컴퓨터공학부

김 민 지

김민지의 공학석사 학위논문을 인준함

2022 년 7 월

위 원 장 _____ (인)
부위원장 _____ (인)
위 원 _____ (인)

Abstract

Bubbleu: Empirical Design Study of Object Detection as an Augmented Reality Game Interaction Method

김민지
Department of Computer Science and Engineering
College of Engineering
The Graduate School
Seoul National University

Object detection with AI can seamlessly integrate user contexts of the physical world into the game context enabling tangible interactions in Augmented Reality (AR) games. AI-based object detection, however, is fundamentally uncertain and erroneous. This paper aims to understand the effects of the object detection uncertainty on the gameplay experience and explore ways to preserve a better gameplay experience despite the errors. I develop Bubbleu, a novel AR pet breeding game that adopts AI-based object detection as a key interaction method. The player manipulates real-world objects to interact with the virtual pet that reacts to the recognized objects inside the game world. Through the empirical design, improvement, and user study process of Bubbleu, I construct and study three game design factors - ambiguity, narrative, and randomness - that possibly improve the gameplay experience when the player faces uncertainty in the object detection based game interaction. I expect that the results of the study reveal design implications for future AR games that provide insights into the adequate game

designs to handle uncertain interactions.

Keywords: Computer Games, Augmented Reality, Object Detection, Human-AI Interaction, Design Study

Student Number: 2020-27351

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Augmented Reality(AR) games, such as Pokemon Go [1] or Angry Birds AR [2], have been steadily gaining popularity over decades. Pokemon Go recorded over 1 billion downloads since its release in 2016, ranking 2nd of all mobile games. Diverse market researches project that the market size of AR games would reach $31.7 billion by 2028 [3] which would be accelerated by more AR glasses hitting the market in the close future [4, 5]. Seamless integration of the game context into the real world boosts the immersion of the augmented game contents [6, 7].

Recent advance in computer vision and deep learning enables a more sophisticated and diversified way to integrate the real world and the virtual game context. State-of-the-art deep neural network(DNN)-based object detection models can detect and recognize objects from in-the-wild (e.g., mobile device camera inputs) images with unprecedented accuracy and can be run on mobile devices in real-time [8–10]. This enables tangible interaction where the players interact with the game world by manipulating real-world objects.

Even before decades, tangible object manipulation as a game mechanic has been ex-

tensively studied in the research community as a proof of concept in a controlled setting. Traditionally, physical markers installed on objects were used for interactions so that they could be automatically detected during the actual gameplay. Despite its confined setting, many game design researches showed the effectiveness of using tangible object manipulation in terms of immersiveness and high-quality user experience [11,12]. DNN-based object detection techniques overcome the limitation of marker-based interaction, allowing users to utilize tangible interaction anywhere with minimal pre-installation while preserving the experience of interaction.

DNN-based object detection, however, has its own fundamental limitations; the uncertainties of the output. DNN-based methods are probabilistic and statistical in nature, heavily relying on the distribution of the training data. Even the state-of-the-art deep learning models trained by huge datasets inevitably suffer from unpredictable inference errors. In AR games, varying input scenes due to diverse players' behaviors, situations, and environments lead to high error rates. The errors of the inference often directly have a negative impact on the gameplay, hindering players from interacting with the game as they intended.

These fundamental limitations of user interactions with DNNs have been recently studied in a broader theme under Human-AI Interaction (HAI) [13–16]. Many existing works, however, focus on traditional AI systems such as chatbots, voice agents, or recommendation systems and suggest general guidelines to make it clear what the AI can and cannot do to the user or scope the results to mitigate the level of uncertainty [13]. Here I argue that we should look deeper into handling uncertainties of DNN-based methods in the game context and that we can figure out more specific design implications to further accelerate the adoption of futuristic interaction methods.

This paper aims to answer two research questions: i) how do the errors of DNN-based object detection affect the user experience when adopting this technology as the main game mechanic for AR games, and ii) what game-specific design improvements

can possibly overcome these fundamental technical limitations? To do so, I discuss the findings from a user study based on a customized game prototype that is carefully designed to evaluate the effects of different error types during gameplay and apply design improvements. Prior to building a prototype game for the user study, I first conducted an extensive literature review and empirical analysis of state-of-the-art deep learning-based object detection models to extract a set of design criteria that could also be generally applied to future game designs. Based on the criteria, I present and implement **Bubbleu**, an AR pet breeding simulation game for handheld mobile devices. Bubbleu is carefully designed to reflect the effects of different types of errors and corresponding design improvements. Next, I conduct a research-through-design investigation in a workshop with a group of HCI researchers to ideate various design concepts to handle the uncertainty of DNN-based object detection. Finally, I conducted a user study with the baseline and the improved version of Bubbleu, including three different error handling designs - *ambiguity*, *narratives*, and *randomness*.

Through the design and user study process of Bubbleu, I discover how the different types of errors in object detection appear and affect the game experience. The result of the study implies that a careful game design possibly preserves the quality of game experience when DNN-based uncertain interaction methods are adopted.

The main contributions of this paper are as follows:

- I propose Bubbleu, a novel AR game that explores object detection as a new means of game interaction.
- I systematically characterize the type and frequencies of object detection errors that have negative effects on game experiences and investigate how different object detection errors affect the player experience.
- Through the design study of Bubbleu, I suggest different game design factors that possibly relieve negative player experience when the players face uncertainties in

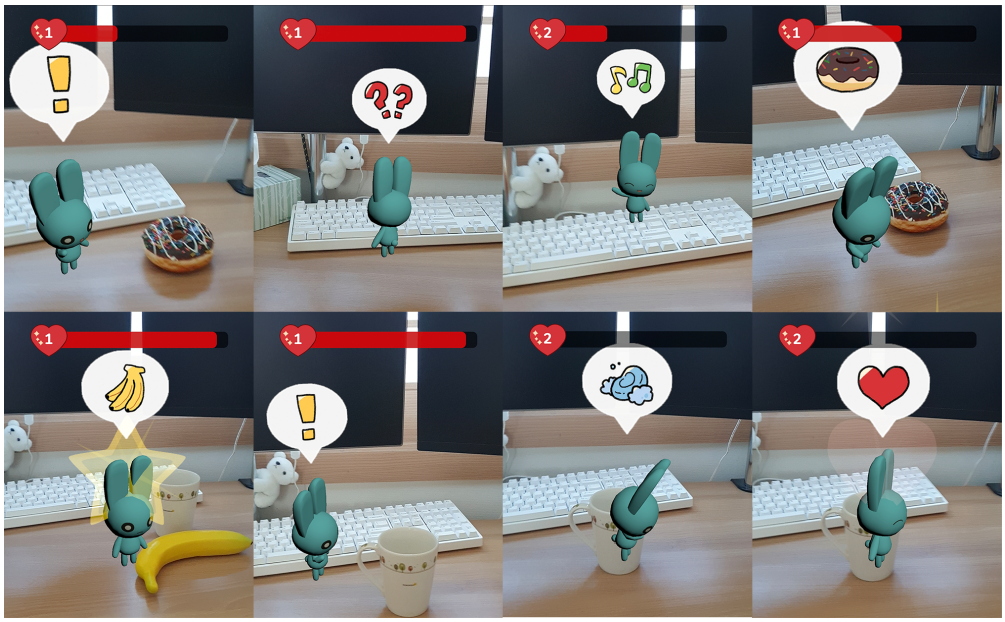

Figure 1.1 Gameplay screenshots of Bubbleu

object detection-based interactions.

Chapter 2

Related Works

2.1 AR Games with Object Detection-based Interactions

There has been a large body of work on AR games with tangible object manipulation interactions. Most of the conventional works utilize marker-based methods for detecting and recognizing interactable objects for the gameplay [11, 12, 17–21]. These games, however, could only be played in fixed settings with a limited set of designated objects which hindered wider adoption of the games. The unprecedented performance achievement of vision AI models (i.e., deep neural networks in computer vision) could overcome the limitations enabling various promising applications in wider settings. AR-Math [22] is an educational application that includes interactions with object detection, enabling users to manipulate real-world objects to interact with the system. Vazquez et al. [23] presented a context-based language learning application utilizing object detection to enhance the efficiency of learning grammar by continuously relating to the real-world context. Despite the improved in-the-wild performance, these applications still suffer from the fundamental limitations of the erroneous and probabilistic nature

of the AI-based object detection model. For example, the user studies in ARMath mentioned that the participants expressed negative experiences when facing the failures of AIs. Motivated by these examples, this paper aims to discover, classify, and analyze different patterns of errors and how they affect the overall user's experience during gameplay.

2.2 AI Errors in HAI

The fundamental limitations of AI errors were long studied in the field of Human-AI Interaction (HAI) [24–26]. There have been efforts to identify the unique challenges of designing AI applications in general and consolidate a guideline to tackle these unique challenges. Amershi et al. [13] proposed 18 guidelines for designing human-AI interactions, 5 of which contain guides to improve user experiences when errors occur in the AI system. Yang et al. [15] examined challenges and solutions in designing human-AI interactions, picking out *capability of uncertainty - uncertainties surrounding what the system can do and how well it performs* - as one of the main challenges of human-AI interaction design. These researches mainly show that (i) it is important to make it clear to the user what the AI-based application can and cannot do and (ii) should provide sufficient error handling designs to maintain user experience quality even when unexpected errors occur [13, 15]. Zargham et al. [27] explored the effects of anticipatory error handling methods in voice-controlled video games. Most of the aforementioned works focus on conventional AI-based applications such as recommendation systems or chatbots. Along the lines of these works, this paper discovers limitations and design implications for errors, specifically in the game context.

2.3 Object Detection Error Analysis

Commonly, mean average precision (*mAP*) analysis based on Intersection over Union(*IoU*) is a representative metric to evaluate the accuracy of object detection [28, 29]. However, *mAP* alone cannot measure the occurrence of different types of errors that appear and affect the user experience of the applications [30, 31], and cannot reveal the temporal nature of sequential frames in video object detection [32]. One of the first research to classify object detection errors into the types of appearance was done by Hoiem et al. [30]. Hoiem et al. analyzed false positive errors of the image object detection errors into different categories, considering the characteristics of the errors and types of the target images. *TIDE* framework [31] proposed by Bolya et al. further developed the error type analysis of object detection, distributing false positive and false negative errors into six types, based on the relation of the detected bounding box and the ground truth. In this work, I classify different types of errors in terms of game experience and study how the different types of errors affect the gameplay with object detection interactions.

Chapter 3

Pre-Design Phase

3.1 Pre-Design Methodology

Prior to the actual game design, I investigate the following factors that are related to object detection interactions in the game: i) how object detection can be used as a game interaction and ii) what potential types of errors occur in gameplay when using object detection as an interaction method. I first analyze existing literature related to object detection-based games(both deep learning-based [22, 33] and non-deep methods [7, 19, 21]). Through the literature review, I aim to understand how object detection is used as a game interaction and if there were any reported errors or discomfort due to the defects of the object detection technology. Then, extending the object detection error analysis in prior works [30, 31], which categorize different types of object detection errors with multiple state-of-the-art deep learning-based object detection models [8–10], I classify errors in object detection into four types. The different impacts of the errors on the game interaction are regarded for classification. Finally, I collect a set of test videos from various settings(e.g., types of objects, occlusions, lighting, camera motion

blur) that are known to affect the accuracy of deep learning-based object detection models([30, 34–38]) and measure the frequency and impact of different types of errors.

3.2 Gameplay Video Analysis

I recorded a total of 530 seconds long videos, including various environmental settings. Table 3.3 describes the number of frames and details on the environmental features of each video set. I extracted a total of 1998 frames from the recorded video in 5 fps units(excluding frames in which no object exists) and performed object detection using the YOLOv3 [39] model trained by the COCO object detection dataset [40]. A total of 11 objects were included in the videos (apple, banana, orange, donut, bottle, cup, book, scissors, lemon, ball, baseball cap). All videos are recorded by a user sitting in front of a wooden desk and holding a smartphone rear camera.

In order to quantitatively classify and analyze each error type, I use the *Intersection-of-Union*(IoU) value of the detected bounding box, which is the most commonly used metric for evaluating the performance of object detection models. The formal definition of IoU is as follows:

$$IoU(P_{bbox}) = \frac{AreaofOverlap(P_{bbox}, GT_{bbox})}{AreaofUnion(P_{bbox}, GT_{bbox})} \tag{3.1}$$

where P_{bbox} is the model's predicted bounding box, GT_{bbox} is the ground truth bounding box, $AreaOfOverlap$ calculates the size of the intersecting area between the two bounding boxes, and $AreaOfUnion$ calculates the size of the union. Larger IoU values represent that the predicted bounding box is similar to the ground truth bounding box, meaning that the localization accuracy is high. More specifically, two constants are used to compare IoU, including foreground threshold(t_f) and background threshold(t_b). When the IoU value is greater than or equal to t_f, it implies that the location of the bounding box area is detected correctly. If the detected result is classified

correctly, the entire detection is meant to be correct. Otherwise, when the detected label is wrong, the result is classified into *Misclassification* error. When the IoU value is less than t_f but greater than t_b, the object is detected successfully but at an invalid location (*Mislocalization*). When the detection occurred but the IoU value is smaller than t_b, it means that the background area is detected as an object(*False detection*). If the detection result of the ground truth object does not exist at all, it means that the detection of the object has failed(*Missing detection*). The constant values $t_f = 0.5$, $t_b = 0.1$ are chosen as same as previous works [30, 31]. Note that *Mislocalization* error and *Misclassification* error can occur at the same time when the classification result and the detected location are both incorrect.

3.3 Design Criteria

3.3.1 Object Detection as Game Mechanics

The key features of object detection technology is **localization** and **type recognition** of detected objects within the given input image(e.g., camera frames). Conventional games that use marker-based object detection as a game mechanic require explicit designation of objects that needs to be localized and recognized by the game device prior to the actual gameplay. With deep learning-based computer vision techniques applied, the objects no longer have to be explicitly identified. Instead, they are recognized as object **classes**; which are defined based on common consensus. For example, ARMath [22], which is a gamified educational app for math learning, uses object detection to generate problems for math concepts(e.g., divide eight batteries into two groups, count the number of chocolates by moving them to virtual trays). The game system is required to recognize the objects as target categories and track the location of the object when the players interact with them. Draxler et al. [33] present a grammar-learning AR application that uses object detection to detect objects around the user and places labels

according to the object class type. Also, it includes a user interaction feature utilizing the object localization property of object detection. When the user moves the objects around, sentences that represent the spatial location of the objects are automatically generated (e.g., the apple is next to the keyboard [33]).

Interaction design utilizing deep learning-based object detection allows the user to use the application anywhere with nearby real-world objects. However, it does not necessarily mean that the app would support interactions with any possible object. Most games choose a set of object classes to be included in the gameplay for user interaction(e.g., 20 object classes in [33]). I define these objects as *interactables* and use the term hereafter. In summary, as of now, object detection as a game mechanic uses its localization and recognition capability for the game to automatically detect **interactables**, represented as common class types, which induces some user behavior as a part of the gameplay.

3.3.2 Taxonomy of Error Types in DNN-based Object Detection

Next, I classify different types of errors that occur in deep learning-based object detection during gameplays. First, I identify the error types of deep learning-based object detection models as shown in Table 3.1. Based on the previous works [30, 31], max IoU value of the bounding box that overlaps the ground truth(IOU_{max}), bounding box of the ground truth, and foreground and background IoU threshold(t_f, t_b) are considered to classify different error types. Although the significance of each error type can be alleviated with further technological advancement, many of the errors are due to the fundamental limitations of the deep learning-based method, that DNN inference is inherently probabilistic [41].

I then reinterpret the identified error types within the game context from the perspective of the player's perception considering the required features of object detection-based games as summarized in the previous section. The final taxonomy of object detection

errors in gameplays is shown in Table 3.2. In the game interaction, perception of the error largely depends on whether the object is interactable or not. Based on the insight, I divide previous types of errors into detailed error types, considering whether the detected object and the ground truth are interactable. For *Missing detection* error type, the player will perceive the error during gameplay only when the detection for an interactable is missing. If the object is non-interactable, the success or failure of the detection does not affect the gameplay at all. Similar analogy can be applied for *False detection* and *Mislocalization* error types. For *Misclassification* errors, the errors can be perceived in several different ways. If the game system classifies an interactable object as another type of interactable object, the game state will change to a different one that the player has not intended. If an interactable object is classified as a non-interactable object, there would be no change in the game state even when the player expected it to. Lastly, when a non-interactable object is recognized as an interactable object, the game state would change without the intention of the player. This interpretation shows that users may not be able to differentiate some types of errors as they generate the same effect in the gameplay. For example, *Missing detection* of interactables and *Misclassification* of interactables as non-interactables are perceived as missing the intended interaction of the player. *False detection* of interactables and *Misclassification* of non-interactables as interactables are perceived as unintended interactions and game state changes during the gameplay.

Lastly, to better understand the significance of the error types, I conducted an analysis based on a set of collected videos. Table 3.4 shows the measured occurrence frequency of different error types on the collected video data. All types of errors occur in most of the settings, but the frequency of the errors varies. The most common error type in all environment settings was *Missing detection* error. *Mislocalization* errors occurred more frequently in environments where occlusion frequently happens(*occluded objects*) and where the camera moves fast(*fast camera movement*). In all settings, *Misclassifica-*

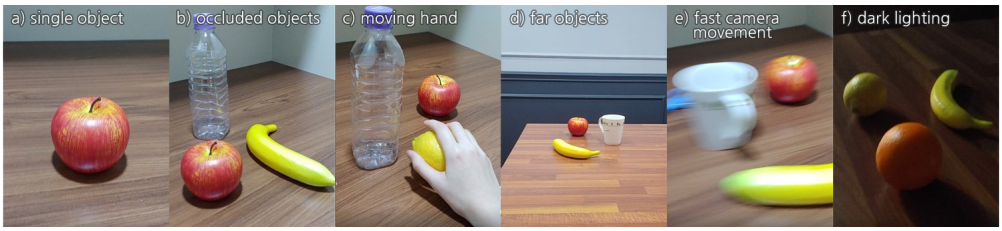

Figure 3.1 Snapshots of recorded gameplay videos in different environment feature settings

tion errors occurred less compared to other types. Among the different video settings, the overall accuracy of *occluded objects* and *fast camera movement* was the lowest. It is notable that when an object is not occluded or distorted by the environments(*single object*), the accuracy of the object detection is considerably high(96.124 %).

Table 3.1 Types of deep learning-based object detection errors

	Error Type	**Description**	**Metric**
Algorithmic Errors	Missing Detection (False Negative)	Fails to localize and recognize the object in the scene	object existing in the scene is not detected at all
	False Detection (False Positive)	Localize and recognize non-existing object	$IoU_{max} <= t_b$ for GT
	Mislocalization	Correct recognition of object in the scene but in wrong location	$t_b <= IoU_{max} < t_f$ for GT
	Misclassification	Correct localization of object in the scene but recognize as different class	$IoU_{max} >= t_f$ for GT of the incorrect class
Systematic Errors	Detection Delay	Latency of processing the detection for current input camera frame	object detection model inference time(ms)
	Device Thermal and Battery Issues	High device temperature or fast battery consumption	device temperature and power consumption

Table 3.2 Reinterpreted Taxonomy of deep learning-based object detection errors in gameplay context

Error Types	Description	User Perceived Error Type
Missing Detection of Interactables	Game system fails to detect an interactable object.	Missing Intended Interaction.
False detection of Interactables	Game system detects non-existing objects as interactables.	Unintended Interaction
Mislocalization of Interactables	Interactables is detected at an inaccurate location.	Wrong location of Intended Interaction.
Misclassification of Interactables as Other Interactables	Classifies detected object as another type of interactable.	Unintended Interaction
Misclassification of Interactables as Non-Interactables	Game system fails to detect an object as non-interactable.	Missing Intended Interaction
Misclassification of Non-Interactables as Interactables	Game system detects non-interactable objects as interactables.	Unintended Interaction.

Table 3.3 Different environmental settings of recorded gameplay environment videos used for error analysis

Name	Total camera frames	Environmental features
Single object	387	Only one object is included in the camera in each frame
Occluded objects	405	Several objects are occluded by each other
Moving hand	488	A hand moves in front of the camera, manipulating and occluding the objects
Far objects	214	Objects are far (about 2 meters) away from the camera
Fast camera movement	220	The camera moves around fast, causing motion blurs
Dark lighting	284	The light of the room is turned off
Total	1998	

Table 3.4 Rates of each error type related to different environment settings in recorded video analysis

Setting	No Errors	Missing Detection	False Detection	Mislocalization	Misclassification
Single Object	96.124	3.618	0	0	0.258
Occluded Objects	40.747	33.604	11.418	1.732	12.5
Moving Hands	53.663	28.515	13.069	1.3867	3.366
Far Objects	56.915	36.968	2.66	3.458	0
Fast Camera	36.875	27.657	19.062	1.719	14.688
Dark Lighting	69.758	18.414	10.618	1.075	0.134

Chapter 4

Bubbleu Game Design Phase

4.1 Design Methodology

The game design phase consists of two stages: i) design of **Bubbleu**; a baseline game that includes all the features of object detection-based game mechanics identified in the pre-design phase with minimal object detection error handling techniques and ii) a design workshop for the ideation of various error handling designs for **Bubbleu-Improved**.

The workshop was conducted with 7 participants. All of them are HCI researchers familiar with deep learning and computer vision techniques. The participants first had a trial session of Bubble-base to understand the game features. Then, they had an hour-length brainstorming session to collect as many ideas as possible for error handling. In the first half-hour, free brainstorming was allowed, and for the second half, the participants were given a set of design guidelines. The guidelines included game design theories which include general game design principles [42–45], and Human-AI guidelines which provides idea on how to handle errors for general AI applications

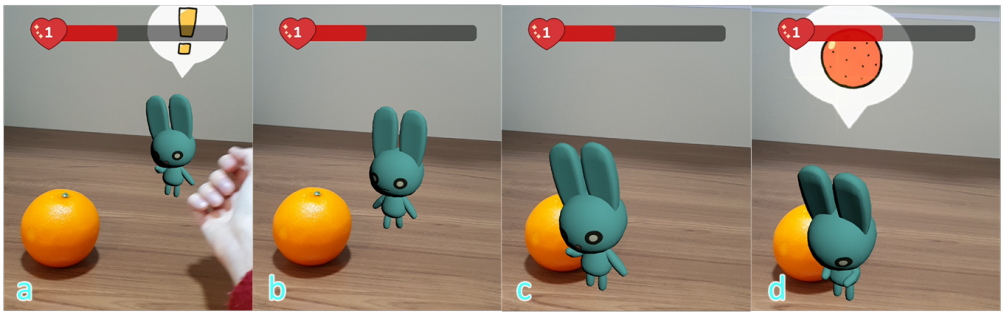

Figure 4.1 The gameplay state sequence of Bubbleu

such as chatbots or recommendation systems [13, 15]. After the brainstorming session, all participants clustered the idea through an affinity diagramming process and agreed on three distinct concepts: *ambiguity*, *narratives*, and *randomness*.

4.2 Overall Gameplay

I first provide the description of Bubbleu gameplay and how it is implemented to meet the design criteria.

In Bubbleu, the goal of the game is to breed a virtual pet. The pet has three needs, eat, wash, and play. Players fulfill the pet's needs by interacting with objects from the physical surroundings. The game device(i.e., handheld smartphone device) uses the rear camera lens to capture the scene. For each frame, a deep learning-based object detection model detects objects within the captured scene. When the gameplay initiates, the pet moves around the planes (e.g., floors or desks) of the real world. The player's role is to bring proper objects of the real world into the scene near the virtual pet to fulfill the pet's needs. For example, the player would place a real apple in front of the virtual pet. When it is detected, the pet will eat the apple and increase its satisfaction gauge. When the satisfaction gauge for three needs is properly fulfilled, the player is rewarded with '*love*' scores, which increases the level of the pet.

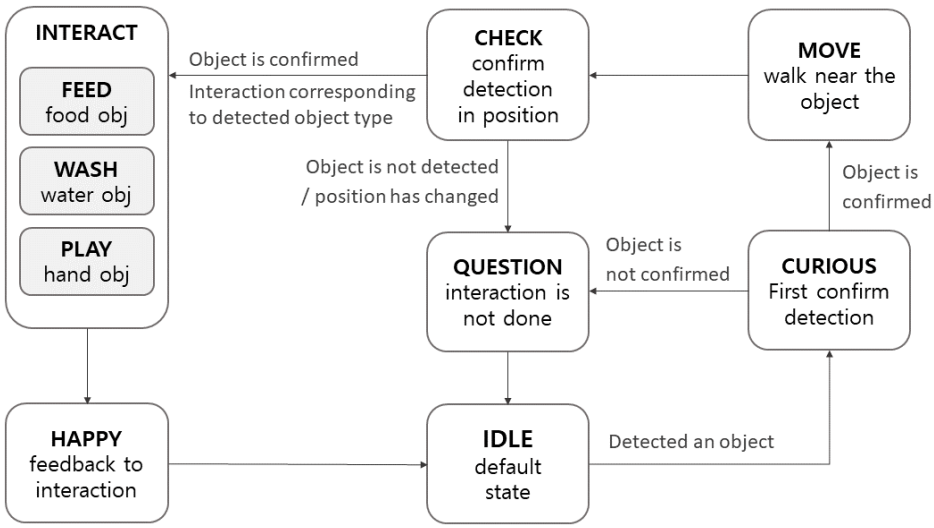

Figure 4.2 The Finite State Machine(FSM) architecture of Bubbleu system.

4.3 Game Finite State Machine(FSM)

Now I provide the details of how Bubbleu is played based on its finite state machine(FSM) to clarify how it meets the design criteria. The FSM also breaks down game design into simpler steps for error handling design improvements. Figure 4.2 shows the full FSM of Bubbleu, which consists of 7 states. The initial and default state of the game is IDLE, where the virtual pet stands at a fixed location.

When the game system detects an interactable in the camera scene, the pet moves into the *curious* phase. In this phase, the pet indicates its curiosity with animation while the system identifies the class of detected objects. When the object is detected and recognized as an interactable for 10 consecutive frames, the pet successfully turns into the *move* phase and walks near to the object. After the pet arrives near the object position, *confirm* phase is executed, checking if the detected object is still in the right position for another 10 consecutive frames. Successful confirmation leads to *interact*

Table 4.1 Types of Interactable Objects and the consecutive interactions in Bubbleu

Interaction Type	Interaction of the Pet	Objects
Food	*The pet eats the object*	*apple*, *banana*, *orange*, *donut*, *carrot*, *broccoli*
Wash	*The pet washes its face*	*cup* *bottle*
Hand	*The pet follows the hand* *The pet is touched*	*hand*

phase where the virtual pet performs different interactions depending on the class of the detected object. If either the curious phase or confirm phase fails, the state first goes to question, which shows an animation of the virtual pet being confused, and returns to the IDLE state, preparing for the next detection.

Figure 4.1 describes each phase of the interaction. Figure 4.1-a indicates *curious* where the pet becomes curious after detecting an interactable object. Figure 4.1-b shows the pet moving to the position of the detected object. In figure 4.1-c, the pet confirms whether the detected object is still in the same position. Finally, in figure 4.1-d, the pet gets into a *interact* phase and interacts the detected object. Here, the pet shows an eating gesture and expresses an *orange* icon as it has detected an orange.

4.4 Implementation Details

As shown in Table 3.1, the performance of object detection does include not only detection accuracy but also other factors such as inference latency or device thermal issues. Several basic error-handling designs are added to the interaction process to minimize the effects of error types unrelated to the purpose of this work.

First, Bubbleu confirms the detection only after checking that the object is consis-

Table 4.2 Validation of design criteria in Bubbleu design

Design Criteria	**Bubbleu system behavior**
Localization capability	The pet detects the location of the object and moves near the object before the interaction.
Recognition capability	The pet detects the class of the object, and indicates the class type by an icon and the animated gesture.

tently detected for 10 consecutive frames in *curious* and *confirm* states. This design is based on the observation that (i) most of the errors appear only in a few frames affected by motion blurs and temporary occlusions, and (ii) playing the pet's animation ensures time to check multiple frames before showing the actual interaction. By referring to multiple frames to confirm the detection, Bubbleu's interaction error rate is decreased low enough to enable the basic interactions.

Secondly, interactable object classes for Bubbleu are selected carefully, considering the model's detection performance. Table 4.1 shows the types of interactable and the consecutive interaction types in Bubbleu. I chose objects that have a clear and characteristic appearance as interactable according to the video analysis of section 3.2 while excluding the others. For example, *apple* and *banana* are recognized with high accuracy even in worse environment conditions including dark lighting or small object size. In contrast, when trying to detect *ball* and *scissors* in such conditions, the model produced errors in many frames.

Lastly, I provide a *'Report'* button on the bottom of the game screen to improve the experience after facing errors by allowing correction of the errors [13]. The players are guided to press the button when they perceive errors during the gameplay. The players can then send the report log of the error situation to the server, which can be used to improve the object detection model of the game in the future.

4.4.1 Design Criteria Validation

Table 4.2 shows a summary of how Bubbleu is designed to meet the design criteria presented in Section 3.3. The localization feature of object detection is used in the *move* phase, where the virtual pet has to move to the nearby location of the detected object to perform the interaction (criteria 1). If the *Mislocalization* error happens, the player can notice it during the *move* phase. The recognition of interactable class is used in the interaction phase, where the detected objects are classified into three interaction types, food, wash, or play (criteria 2). In this phase, *Misclassification* error can occur and be noticed. On the other hand, *Missing detection* and *False detection* errors appear in *idle*, *curious* and *confirm* phase, where the player can notice if the intended interaction is not occurring or an unintended interaction is occurring.

Chapter 5

Error Improvement Designs

Even with the efforts to minimize the effect of the errors, it is impossible to fundamentally prevent errors from occurring and affecting the gameplays. Through the Bubbleu game design workshop described in Section 4.1, three concepts that are widely adopted in games and general AI application designs - *ambiguity*, *narratives*, and *randomness* - were decided. The design improvements aim to improve the gameplay experience when an unexpected error occurs during the gameplay.

Ambiguity is a concept that has been addressed in diverse HAI studies related to the uncertainty of AI inferences [13, 15]. Amershi et al. presented a guideline to scope services when the system is uncertain about the inference result, such as providing multiple choices to the user in an AI inference-based suggestion system.

Bubbleu implements the scope of uncertainty by grouping multiple interactable objects that are easily misclassified from each other into the same interaction, instead of showing detailed classification results as an icon. For example, as an *apple* and an *orange* have a high chance of being wrongly classified into each other, the two objects are expressed as the same *'eat'* interaction. Similarly, a *bottle* is often misclassified

to a *cup*, so the resulting interaction of the two object is mapped to a same *'wash'*. In this design, players cannot distinguish between the result of the detection in the same interaction group. This design has a tradeoff, as it prevents the *Misclassification* from being perceived by the player but also sacrifices the variety of interaction.

In addition to the label expression, the precision of the location of the *move* phase is also lowered in the *ambiguity* design. I expected to improve the *Mislocalization* error by adding ambiguity to the perceived location of the detection.

Narrative is one of the essential components of the game. Diverse game and gamification researches have been focusing on the effects of narratives on game experiences in terms of aesthetic enjoyment and emotional fulfillment [42, 43, 46, 47]. Bubbleu examines the potential of narratives to give an intuitive and persuasive explanation to the uncertain game system.

I embed *Narrative* error improvement component to Bubbleu in two ways. First, when starting the game, an additional prologue that shows the background story of the pet is introduced as a short cartoon. The prologue explains the uncertainty of the system as *'this pet was born yesterday and sent to Earth so that it can sometimes confuse several objects'*, shifting the responsibility of errors occurring during the gameplay to the immaturity of the pet. Second, *'Report'* button to fix the error is substituted by a *'Teach'* button. While the function of the substituted button is the same as the original, the guidance to the player is converted to *'teach the pet about the objects it confuses and fix the behavior of the pet'* instead of *'report the error and improve the DNN model'*.

Randomness is another widely used design component in many commercial and non-commercial games [45]. Designers often choose to add random features to add diversity and surprising experience to the game, from giving random rewards to the same task to generating random levels for each gameplay. I also come up with an idea to add randomness to the behavior of the pet.

The random behavior of the pet gives perturbation to the basic interaction behaviors.

In the *idle* phase, the pet sometimes moves to a random position and shows curiosity when no interactable is detected for a while. Also, it sometimes ignores the object and wanders around even if an interactable is detected around. This random behavior relieves *Missing detection* errors, as the pet shows continuous feedback to the player when the intended interactable is not being detected. The pet's random behavior has an additional advantage in that the wandering behavior makes the pet look more natural as a real creature, compared to the original behavior of always standing still in the *idle* phase.

Chapter 6

User Study Phase

I conducted two user studies to understand how the Bubbleu game design affects the user experience under the error circumstances. The user study consists of two sections: (i) a controlled study comparing design factors with manually injected errors and (ii) a real gameplay study with the full improvements. First, I investigate how the different improvement designs of Bubbleu affect the player experience when the game interaction produces errors. Second, I conduct a qualitative in-the-wild gameplay study to understand the users' game experience and perception of errors in the real gameplay situation.

6.1 Controlled Study with Injected Errors

6.1.1 Method

I created three different improved version of Bubbleu, each implementing the error improvement design suggested in Section 5(i.e., *Bubbleu-ambiguous*, *Bubbleu-narrative*, *Bubbleu-random*) in addition to the original Bubbleu(i.e., *Bubbleu-baseline*) as a base-

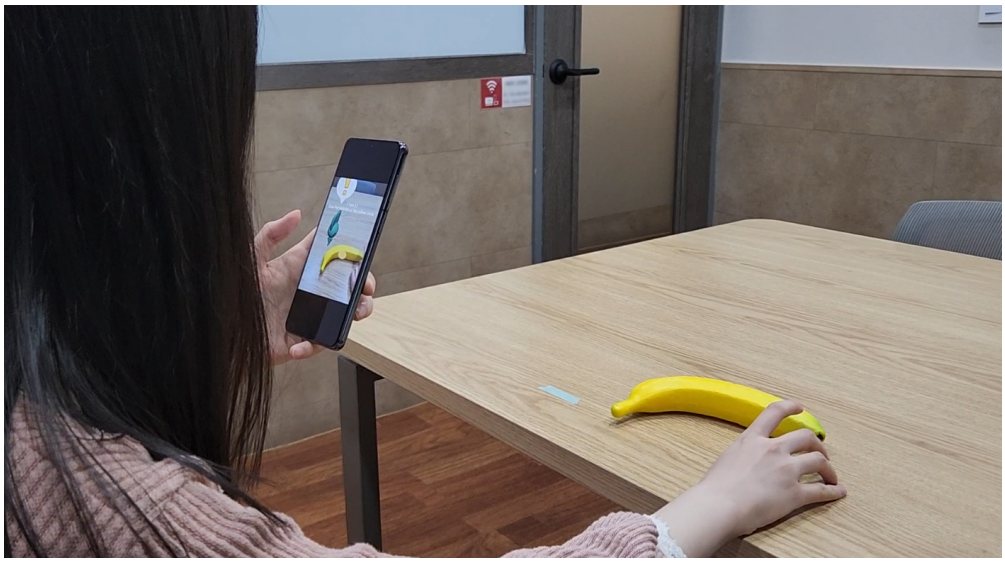

Figure 6.1 Wizard-of-Oz gameplay study with injected errors.

line. 36 participants were recruited through online surveys. The participant group consists of 19 males and 14 females, all in the age group between 20 and 29. According to the survey, 4 of them reported having no experience of using AR applications such as Pokémon Go [48]. The others reported that they have prior experience of using AR applications. 16 of them were unfamiliar with the object detection techniques, and the others had prior knowledge related to object detection. The experiment consists of two sessions and a comprehensive interview. Each session includes a 20 minutes gameplay of Bubbleu, one randomly chosen improved version, and the baseline(i.e., Each design factor was tested with 12 participants).

The experiment was conducted in a Wizard-of-Oz method. To control the occurrence of different types of errors during the interaction, I designed the Wizard-of-Oz experiment not to allow free interaction. Instead, it only consists of 10 sequential interaction tasks. Out of 10 tasks, 4 different types of errors including *Missing detection*, *False detection*, *Misclassification*, and *Mislocalization* are injected in 4 random tasks.

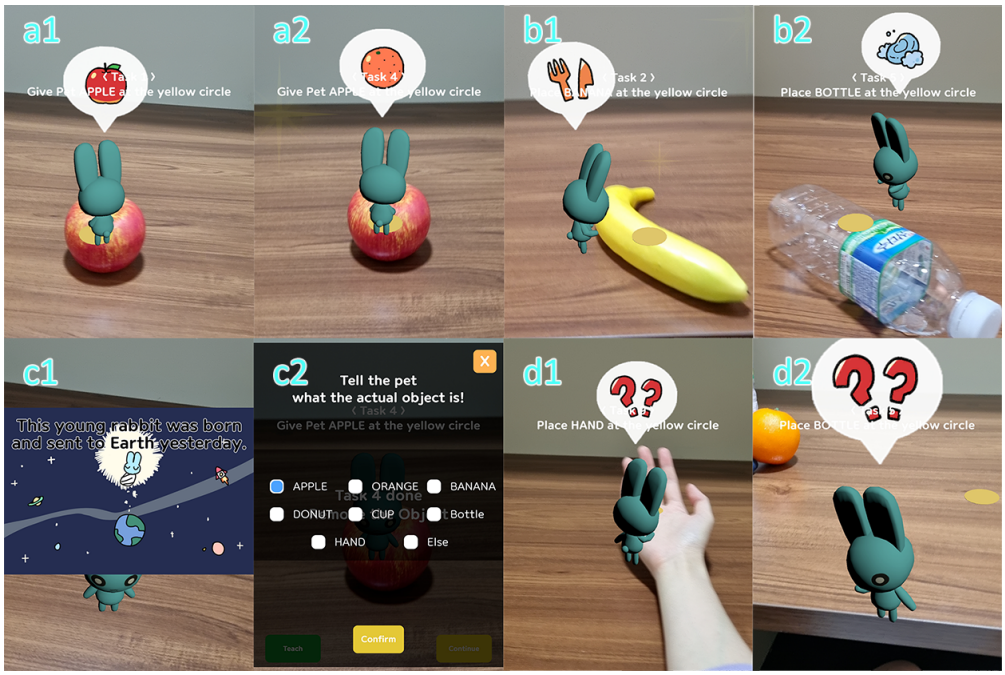

Figure 6.2 Snapshots of four different Bubbleu designs during controlled user study

The other 6 tasks do not include any errors.

Figure 6.1 describes how the original Bubbleu and the improvement designs are implemented for the controlled gameplay experiment. Common tasks are manipulated to succeed in interaction with no error, regardless of the actual detection results(Figure 6.1-a1). Tasks with injected error use the detected results, which are manually perturbed to have different types of error(Figure 6.1-a2 - The pet shows *orange* icon while the ground truth object is an *apple*). For details, see Table 6.1. Figure 6.1-b1 and b2 show interactions in *Bubbleu-ambiguous*. In *Bubbleu-ambiguous*, the interaction type is shown as an icon instead of the object type, and the precision of the movement range is set larger than the original Bubbleu. Figure 6.1-c1 and c2 indicate additional narrative components in *Bubbleu-narrative*; each representing prologue cartoon and *teach* window. Figure 6.1-d1 and d2 show random perturbation behaviors in *Bubbleu-*

Table 6.1 Tasks with injected errors of the controlled user study

Error type	**Appearance**
None	Task is completed with no error
Missing detection	The interaction does not happen until the end of the task
False detection	Random interaction happens at random location before the player finishes the task
Misclassification	The interaction happens but the type of the interaction icon is different to the ground truth
Mislocalization	The interaction happens but the location of the interaction is different to the ground truth

random. In random tasks, the pet wanders around and shows curiosity, ignoring the interactable.

During each session, the participant is guided to perform 10 different tasks. Six objects to interact are provided, including *apple, banana, orange, donut, mug*, and *bottle*. Each task requests the participant three sequential operations using a specific target object: i) moving the object to the indicated position on the desk, ii) observing the interaction between the pet and the object, and iii) removing the object from the desk. The message on the game UI instructs the participant to perform each operation.

The participants are instructed to press the *"report"* button (*"teach"* button instead in *Bubbleu-narrative*) on the screen before proceeding to the next task when they have perceived the pet behaved differently from their expectations. At the end of the experiment, I interviewed them about how they perceived the errors, referring to the recorded gameplay videos and the error report logs.

Table 6.2 Perceived rate of each error type for each Bubbleu design version in the controlled study

Played version	**Missing detection**	**False detection**	**Misclassification**	**Mislocalization**	**Total rate**
Bubbleu-Baseline	0.83	0.89	0.61	0.06	0.60
Bubbleu-Ambiguous	0.67	0.83	0	0.17	0.42
Bubbleu-Narrative	0.83	0.83	0.50	0	0.54
Bubbleu-Random	0.83	0.67	0.67	0.17	0.58

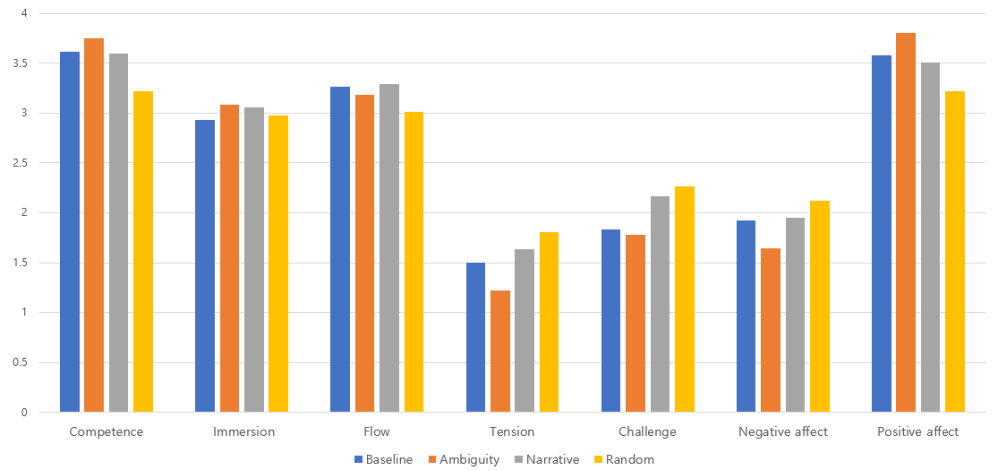

Figure 6.3 Game Experience Questionnaire results

6.1.2 Results

Perception of errors. After each gameplay session, two questions related to the perception of errors were asked - *"Which type of error did you perceive to have happened?"* and *"Which error was the most frustrating?"*. For the first question, *Missing detection* and *False detection* error turned out to be perceived by almost every participant in every gameplay session, whereas *Mislocalization* error was almost not perceived as an error. More than half of the participants answered that they had perceived *Misclassification* error during the gameplay, except for those who played *Bubbleu-ambiguous*. As *Bubbleu-ambiguous* hides the precise labels of the objects, *Misclassification* of the objects in the same interaction type is not revealed to the player.

For the second question, 6 participants(P1, P3, P4, P6, P11, P16) answered that the *Missing detection* error was the most frustrating. Another 6(P2, P8, P9, P13, P17, P18) picked out error as the most frustrating. 2 participants(P5, P14) even stated that those two errors were both the worse. 2 participants(P7, P15) answered that *Misclassification* was the most frustrating, and the remaining one(P12) chose *Mislocalization*. Several

participants added that *"Although I pointed out the most frustrating errors, it was not that frustrating, and I could enjoy the game"*(P5, P10, P15).

Effects of design improvements. Participants answered the game experience questionnaire(GEQ) [49, 50] after each session(6.1). I also asked how the participant's experience and preference differed between the two continuous game sessions. For the GEQ results, *Bubbleu-ambiguous* scored highest value in *Positive affect* component and lowest value in *Negative affect* and *Tension* component. Among 12 players who played *Bubbleu-ambiguous*, 8 participants(P2, P5, P9, P10, P18, P22, P24, P27) indicated that *Bubbleu-ambiguous* had lower error rate. 6 participants(P5, P9, P24, P22, P24, P27) answered that they preferred *Bubbleu-ambiguous* to *Bubbleu-baseline* because the game experience was more smooth due to the low error rate. P2 and P35 mentioned that they preferred *Bubbleu-baseline* because it allows wider interactions. In contrast, P10, P18, and P31 commented that they preferred interaction of *Bubbleu-ambiguous* because *"I preferred watching the interaction behavior of the pet than watching it distinguishing object types"*(P10) and *"it allows me to imagine the pet's thoughts by showing behaviors"*(P18).

Bubbleu-narrative scored the highest value in *Flow* component of the GEQ, which indicates the engagement of the game. All participants who played *Bubbleu-narrative* agreed that showing a narrative prologue in *Bubbleu-narrative* gave a better gameplay experience than *Bubbleu-baseline*. P17 addressed that *"the existence of narrative itself makes it feel more like a game"*. Five of them(P1, P15-P17, P29) also mentioned that the existence of *"teach"* button gave nice feelings in that *"I can correct the behavior of the pet by additional interaction"*(P1, P15, P16) and *"focusing on teaching itself feels like another challenge of the game"*(P17). Two participants(P15, P16) mentioned that the detection in *Bubbleu-narrative* was more accurate than *Bubbleu-baseline*, which is actually not true, even though the error reports of themselves in the two games were the same. In contrast, no participant reported that *Bubbleu-baseline* was more accurate than

Bubbleu-narrative. Additionally, 4 participants who weren't given *Bubbleu-narrative* version mentioned that *"it would be nice if you show the stories of the pet during the game"*(P2, P5, P18, P35).

Participants who played *Bubbleu-random* gave diverse answers. P3 answered that they preferred *Bubbleu-random* because it had better accuracy. P6 and P13 answered the opposite - that the detection accuracy of *Bubbleu-baseline* was the better. P11 mentioned that they liked the pet wandering around because *"it makes the pet look more like alive"*. P12 and P14 answered that they did not notice the difference until hearing how the two games were different during the interview. For the GEQ results, *Bubbleu-random* tend to have the worst values in every component except the *Immersion*.

Note that although the random behaviors of the pet were as same as in the real Bubbleu gameplay, the perception of randomness in the experiment appears different from the real Bubbleu. In real Bubbleu gameplay, interactions do not occur continuously. As the need of the pet is satisfied right after the previous interaction, it is impossible to interact with the pet multiple times in a short time period. In this situation, the pet's random behavior between the interaction phases appears relatively natural. In the controlled gameplay, however, the interactions happen consecutively, so the sudden random behavior right after the interaction often confuses the player.

6.2 Real Gameplay Study

6.2.1 Method

12 new participants were recruited through online surveys. All participants had experience of playing other commercial AR applications. Three reported that they were familiar with object detection techniques, while the others did not.

In the experiment, participants play Bubbleu for 10 minutes. I provide six Bubbleu-interactable objects to each participant, including *apple, banana, orange, donut, mug*,

and *bottle*. Using the objects, the participants freely interact with the virtual pet. They can feed the pet using food objects, wash it using a mug and bottle, and call or play with it to a specific position using their hands. The participants play the game until the pet's *Affection Gauge* becomes full. They are also encouraged to interact with as diverse objects as possible.

6.2.2 Results

Errors perceived by users. I observe that the frequency and composition of error occurrences vary significantly across different game players. Some participants quickly learn the ways to make the object detected correctly after moving around the interactable objects several times. In contrast, some participants end up failing to detect some of the objects.

I also asked the participants 'which type of error was the most frustrating?'. Five participants(P1, P2, P5, P7, P8) indicated *Misclassification of an interactable to a non-interactable* and *Missing existing interactable* errors. P1 said *"The interaction I intended was ignored, which made me very frustrated"*. Four participants(P4, P6, P10, P11) said that the *Detecting not existing non-interactable* error was most frustrating. They experienced the *Detecting not existing non-interactable* error in which the pet moved to a far corner of the desk. P6 said *"The pet moving far away seems strange to me"*. One participant(P12) mentioned the *Misclassification of an interactable to another interactable* error to be most frustrating. Two other participants(P3, P9) said that *"Several errors occurred, but they were not that frustrating."* and that *"It was okay. I eventually interacted with the pet as I intended"*. I additionally observe that the player has the opportunity to recover from the *Misclassification of an interactable to another interactable*, *Misclassification of an interactable to a non-interactable*, and *Missing existing interactable* errors by changing the way to hold the object or camera when they can guess the possible causes of the errors. *Detecting not existing non-interactable* has

a different problem since the player does not clearly understand the type and causes of the error.

General gameplay experiences. 11 participants mentioned that the virtual pet of Bubbleu is cute. 10 participants said that they would like to play Bubbleu again. 3 participants(P5, P9, P12) explicitly mentioned that they felt some sort of fun *"trying to make the object detected"* during the evaluation.

Several participants mentioned the expectation of more diverse interactions in the gameplay. P2 asked *"Can I give something to drink to the pet?"*. P12 said *"I want to push the pet with my hand to recognize the object"* when the pet failed to interact with the object. P3 said *"I want to touch and tickle the pet"*. I anticipate that diverse interactions with real-world objects can be explored in different games. Various interactions enrich the game experience, but the uncertainty of the AI system also increases according to the diverse inferences required.

Other observations. I observed that the participants learned the way of making objects detected well. P5 said *"the bottle is well detected when it is upside down"*, and kept placing the bottle object upside down during the whole experiment. Several other participants developed their own strategies of placing objects, such as laying down the *apple* object or keeping to scanning the *donut* object exactly in the upward direction. One participant(P2) figured out that even when the camera position is fixed, the pet eventually interacts with the object after some time passes. Based on the bias, they held the device fixed around the object until the task time was over and the pet was forced to interact with the object.

Several participants(P3, P5, P8, P9, P12) added their own narratives to the behavior of the pet. P3 mentioned that *"I think it does not want to eat the banana"* when the *banana* was not detected for a while. P12 said *"the pet tries to wash with the apple. It seems to prefer washing than eating"*.

Most participants(all except P1, P6, and P7) agreed that they would like it better

to play Bubbleu with head-mount type AR devices than the handheld device in terms of immersion. Those who disagreed mentioned usability(*"AR glasses are heavy and uncomfortable"* - P1, P7) and low accessibility to the AR head-mount devices(*"I would like to play it casually using my smartphones instead of the large devices"* - P1, P6).

Chapter 7

Discussion

7.1 Design Implications for Object Detection Interaction

I discuss the design implications of object detection-based game interactions according to the findings through the design, analysis, and user study results of Bubbleu. Although completely overcoming the inherent uncertainty of object detection is impossible, the design study of Section 3-6 reveals that the negative effects of object detection errors can be potentially relieved by a careful game design. In this section, I explore further design implications based on the findings from the design process and the user study observations.

Precision of the detection results in gameplay. Hiding precise results and showing only metaphors about the result of the AI inference can prevent specific types of errors from being perceived. Instead of displaying detailed inference results, I suggest utilizing signs that indirectly indicate the inference results to the players. For instance, in Bubbleu, the virtual pet walks and stops before a certain distance from the detected object instead of walking right next to the object. The pet's behavior is designed to appear natural

even when there is a certain level of *Mislocalization* errors.

Speed of Interactions. I suggest designing slow and steady game interactions when developing games with AI inferences for two reasons. The first reason is to ensure the AI inference system has sufficient time to interpret real-world data. Bubbleu is intentionally designed with slow-paced interactions, where the scene of the gameplay does not change much, and the players do not have to interact in haste. Bubbleu also reserves time for the AI inference system to detect the scene over multiple frames by adding a few seconds of resting time for the pet between interactions. By deciding object detection results referring to multiple frames, a large portion of *Detecting not existing interactable* and *Missing existing interactable* errors are improved since these errors mainly occur in the camera frames blurred or occluded due to camera movements.

The second reason is to allow sufficient time for the players to interact with the AI inference system correctly and recover from unexpected errors. In Bubbleu, when an intended interaction does not happen due to *Missing existing interactable* or *Misclassification of an interactable to a non-interactable* errors, the player can simply try the interaction again after changing camera angles or moving the object around. If the interaction is done without reserving sufficient time to recover from the errors, the risk of failure is likely to frustrate the players more.

Gamifying user behaviors to overcome the errors. During the user study, I observed that many participants actually 'enjoyed' the actions to overcome the errors when the object is not detected correctly at once - moving the device around and figuring out how to make objects detected well in the system. Several participants showed signs of enjoyment, such as laughing or exclaiming, when they succeeded in the interaction after challenging to detect the object. P7 of the real gameplay study explicitly mentioned that *"I had fun with thinking how I could make the object detected well by the system"*.

Based on the observations, I suggest designing players' actions to recover from the errors to be enjoyable as a part of the gameplay. In Bubbleu, actions including

moving the device around or changing the position of the objects become the gameplay interaction by themselves. Adding enjoyable feedback and reward will change the struggles to overcome the errors into a *'playable'* challenge.

7.2 Generalization of Bubbleu

7.2.1 Generalization to other games using object detection

Object detection is an interaction method that enables tangible object interaction and context recognition in lightweight AR games. Bubbleu used object detection to enable tangible object interaction to breed a virtual pet, such as feeding and washing virtual pets through real food and cup objects. In addition to interactions of Bubbleu, various characteristics of object detection have the potential to deliver diverse game experiences. For instance, ARMath [22] utilized object detection to detect daily objects to interact as tangible tools for math education. Liang *et al.* [51] analyzed the background contexts via object detection and generated suitable virtual contexts to be coupled with the physical environment.

Implications from error analysis of Bubbleu, such as effects of errors on gameplay interaction and effects of environmental features on the occurrence of errors, can be applied as a useful measure over diverse object detection interactions.

7.2.2 Generalization to other AI Inference-based interactions

With the recent improvement in mobile device performance and deep learning models, diverse deep learning-based AI interaction has gained a large potential to be utilized as a game interaction method. Various researchers and developers are trying to deliver novel game experiences using AI inference-based interactions, including hand tracking, eye tracking [52], face tracking [53], and scene detection [51]. However, various AI inference methods cannot ignore the uncertainty problem of AI inferences become a

large challenge as in object detection. Bubbleu's game context-aware error taxonomy and analysis can provide useful directions to analyze errors in other AI inference-based game interactions.

For instance, a game that utilizes vision-based hand tracking as an interaction input will define pre-determined interactable hand gestures distinguished from other non-interactable gestures. Hand detection models, including Mediapipe [54] have high performance when a hand is well visualized, while it is error-prone to false positives when no hand is detected in the scene, or the hand is occluded. Our taxonomy of errors can be similarly applied to analyze how different types of errors occur and affect the gameplay and help design interactions to minimize the negative effects of the errors.

Chapter 8

Conclusion

Object detection has a large potential to be used as a key interaction method in future AR games and applications. However, the inherent uncertainty of object detection makes it difficult to guarantee user experiences in cases of uncontrollable errors. I analyze the pain points of object detection and present Bubbleu, an AR game with an object detection interaction, to investigate the potential of object detection as a game interaction. Through empirical pre-design analyses and the game design process, I analyze possible object detection errors that can have negative effects on the interaction. The continuing design improvement process and user study reveal the effect of game design to improve the user experience with the uncertain interaction mechanics. I further discuss game design implications for object detection interactions in diverse future AR games.

Bibliography

[1] "Pokemon go." [Online]. Available: https://pokemongolive.com/

[2] "Angry birds ar: Isle of pigs." [Online]. Available: https://www.angrybirds.com/angry-birds-ar-isle-pigs/

[3] V. M. Research, "Augmented reality gaming market size and forecast," March 2022.

[4] A. Truly, "Apple's ar headset could be available by june next year," May 2022.

[5] Y. M. Paresh Dave, "Google's second try at computer glasses translates conversations in real time," May 2022.

[6] R. Wetzel, R. McCall, A.-K. Braun, and W. Broll, "Guidelines for designing augmented reality games," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, ser. Future Play '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 173–180. [Online]. Available: https://doi.org/10.1145/1496984.1497013

[7] C. Matysczok, R. Radkowski, and J. Berssenbruegge, "Ar-bowling: Immersive and realistic game play in real environments using augmented reality," in *Proceedings of the 2004 ACM SIGCHI International Conference on Advances*

in Computer Entertainment Technology, ser. ACE '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 269–276. [Online]. Available: https://doi.org/10.1145/1067343.1067379

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," pp. 21–37, 2016.

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[10] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[11] R. Wetzel, R. McCall, A.-K. Braun, and W. Broll, "Guidelines for designing augmented reality games," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, 2008, pp. 173–180.

[12] R. Wetzel, L. Blum, W. Broll, and L. Oppermann, "Designing mobile augmented reality games," in *Handbook of Augmented Reality*. Springer, 2011, pp. 513–539.

[13] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen *et al.*, "Guidelines for human-ai interaction," in *Proceedings of the 2019 chi conference on human factors in computing systems*, 2019, pp. 1–13.

[14] G. Bansal, B. Nushi, E. Kamar, D. S. Weld, W. S. Lasecki, and E. Horvitz, "Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 2429–2437.

[15] Q. Yang, A. Steinfeld, C. Rosé, and J. Zimmerman, "Re-examining whether, why, and how human-ai interaction is uniquely difficult to design," in *Proceedings of the 2020 chi conference on human factors in computing systems*, 2020, pp. 1–13.

[16] K. I. Gero, Z. Ashktorab, C. Dugan, Q. Pan, J. Johnson, W. Geyer, M. Ruiz, S. Miller, D. R. Millen, M. Campbell *et al.*, "Mental models of ai agents in a cooperative game setting," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–12.

[17] Y. Xu, E. Barba, I. Radu, M. Gandy, R. Shemaka, B. Schrank, B. MacIntyre, and T. Tseng, "Pre-patterns for designing embodied interactions in handheld augmented reality games," in *2011 IEEE International Symposium on Mixed and Augmented Reality-Arts, Media, and Humanities*. IEEE, 2011, pp. 19–28.

[18] Y. Xu, S. Mendenhall, V. Ha, P. Tillery, and J. Cohen, "Herding nerds on your table: Nerdherder, a mobile augmented reality game," in *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, 2012, pp. 1351–1356.

[19] C.-e. Lin, T. Y. Cheng, and X. Ma, "Architect: Building interactive virtual experiences from physical affordances by bringing human-in-the-loop," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.

[20] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *Ieee Access*, vol. 5, pp. 6917–6950, 2017.

[21] I. Alakärppä, E. Jaakkola, J. Väyrynen, and J. Häkkilä, "Using nature elements in mobile ar for education with children," in *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*,

ser. MobileHCI '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3098279.3098547

[22] S. Kang, E. Shokeen, V. L. Byrne, L. Norooz, E. Bonsignore, C. Williams-Pierce, and J. E. Froehlich, "Armath: augmenting everyday life with math learning," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–15.

[23] C. D. Vazquez, A. A. Nyati, A. Luh, M. Fu, T. Aikawa, and P. Maes, "Serendipitous language learning in mixed reality," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2172–2179. [Online]. Available: https://doi.org/10.1145/3027063.3053098

[24] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *arXiv preprint arXiv:1703.04977*, 2017.

[25] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," *arXiv preprint arXiv:2002.06470*, 2020.

[26] J. Caldeira and B. Nord, "Deeply uncertain: comparing methods of uncertainty quantification in deep learning algorithms," *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 015002, 2020.

[27] N. Zargham, J. Pfau, T. Schnackenberg, and R. Malaka, ""i didn't catch that, but i'll try my best": Anticipatory error handling in a voice controlled game," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: http://lps3.doi.org.libproxy.snu.ac.kr/10.1145/3491102.3502115

[28] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[29] "Coco analysis toolkit." [Online]. Available: http://cocodataset.org/detection-eval

[30] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," pp. 340–353, 2012.

[31] D. Bolya, S. Foley, J. Hays, and J. Hoffman, "Tide: A general toolbox for identifying object detection errors," pp. 558–573, 2020.

[32] H. Mao, X. Yang, and W. Dally, "A delay metric for video object detection: What average precision fails to tell," 08 2019.

[33] F. Draxler, A. Labrie, A. Schmidt, and L. L. Chuang, "Augmented reality to enable users in learning case grammar from their real-world interactions," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–12.

[34] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury, "Frugal following: Power thrifty object detection and tracking for mobile augmented reality," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, ser. SenSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 96–109. [Online]. Available: https://doi.org/10.1145/3356250.3360044

[35] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," *CoRR*, vol. abs/1812.05050, 2018. [Online]. Available: http://arxiv.org/abs/1812.05050

[36] N. Buch, S. A. Velastin, and J. Orwell, “A review of computer vision techniques for the analysis of urban traffic,” *IEEE Transactions on intelligent transportation systems*, vol. 12, no. 3, pp. 920–939, 2011.

[37] M. Z. Uddin and J. Torresen, “A deep learning-based human activity recognition in darkness,” in *2018 Colour and Visual Computing Symposium (CVCS)*. IEEE, 2018, pp. 1–5.

[38] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” *arXiv preprint arXiv:1905.05055*, 2019.

[39] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” pp. 740–755, 2014.

[41] A. B. Patel, M. T. Nguyen, and R. Baraniuk, “A probabilistic framework for deep learning,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/c70daf247944fe3add32218f914c75a6-Paper.pdf

[42] R. Hunicke, M. LeBlanc, and R. Zubek, “Mda: A formal approach to game design and game research,” vol. 4, no. 1, p. 1722, 2004.

[43] P. Sweetser and P. Wyeth, “Gameflow: a model for evaluating player enjoyment in games,” *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 3–3, 2005.

[44] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, “From game design elements to gamefulness: Defining "gamification",” in *Proceedings of the 15th International*

Academic MindTrek Conference: Envisioning Future Media Environments, ser. MindTrek '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 9–15. [Online]. Available: https://doi.org/10.1145/2181037.2181040

[45] W. L. Bedwell, D. Pavlas, K. Heyne, E. H. Lazzara, and E. Salas, "Toward a taxonomy linking game attributes to learning: An empirical study," *Simulation & Gaming*, vol. 43, no. 6, pp. 729–760, 2012.

[46] D. Robinson and V. Bellotti, "A preliminary taxonomy of gamification elements for varying anticipated commitment," in *Proc. ACM CHI 2013 Workshop on Designing Gamification: Creating Gameful and Playful Experiences*, 2013.

[47] M. D. Dickey, "Murder on grimm isle: The impact of game narrative design in an educational game-based learning environment," *British journal of educational technology*, vol. 42, no. 3, pp. 456–469, 2011.

[48] Niantic, "Pokémon go." [Online]. Available: https://www.pokemongo.com/en-us/

[49] W. A. IJsselsteijn, Y. A. De Kort, and K. Poels, "The game experience questionnaire," 2013.

[50] K. L. Norman, "Geq (game engagement/experience questionnaire): a review of two papers," *Interacting with computers*, vol. 25, no. 4, pp. 278–283, 2013.

[51] W. Liang, X. Yu, R. Alghofaili, Y. Lang, and L.-F. Yu, "Scene-aware behavior synthesis for virtual pets in mixed reality," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445532

[52] A. Ramirez Gomez and H. Gellersen, “More than looking: Using eye movements behind the eyelids as a new game mechanic,” in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, 2020, pp. 362–373.

[53] R. B. Robinson, E. Reid, J. C. Fey, A. E. Depping, K. Isbister, and R. L. Mandryk, “Designing and evaluating’in the same boat’, a game of embodied synchronization for enhancing social play,” in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–14.

[54] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “Mediapipe hands: On-device real-time hand tracking,” *arXiv preprint arXiv:2006.10214*, 2020.

초록

미래형 증강 현실(Augmented Reality, AR) 게임을 개발할 때, 인공지능(Artificial Intelligence, AI)에 기반한 객체 인식(Object Detection) 기법을 활용하면 현실 세계에 있는 사용자의 환경을 해석해 가상의 게임 환경을 통합하여 보여줌과 동시에 현실의 유형 물체(Tangible Object)를 자연스럽게 활용하여 게임을 조작하는 것을 가능케 함으로써 높은 몰입감을 동반한 우수한 게임 경험을 전달할 수 있다. 그러나 인공지능 기반의 객체 인식 인터렉션은 결과가 불확실하고 오류가 발생하기 쉽다는 근본적인 문제가 있다. 본 논문에서는 객체 인식의 불확실성이 게임 경험에 미치는 경험을 이해하고, 그러한 불확실성을 직접 극복할 수 없더라도 게임의 특징을 활용한 설계를 통해 에러가 발생했을 때의 사용자 경험을 향상시키는 방법을 탐구하고자 한다. 우선 본 논문은 Bubbleu라는 이름의 객체 인식 인터렉션을 활용하는 펫 키우기 시뮬레이션 증강 현실 게임을 제안한다. Bubbleu에서 사용자는 현실 물체를 조작하여 가상의 펫 캐릭터와 상호작용한다. 본 논문의 연구는 Bubbleu를 설계하고, 개선하고, 사용자 연구(User Study)를 실행하는 과정으로 이루어진다. 이러한 경험적 연구 과정을 통해 객체 인식 기반 인터렉션을 활용하는 게임에서 에러가 발생했을 때 사용자 경험(User Experience)이 저하되는 것을 방지할 수 있는 3가지의 게임 설계 요소인 모호성(Ambiguity), 내러티브(Narrative), 무작위성(Randomness)을 제안한다. 본 논문의 연구 결과를 통해 미래 증강 현실 게임에서 객체 인식 기반 인터렉션을 활용하고자 할 때 활용할 수 있는 게임 설계 방법론을 제시하고자 한다.

주요어: 컴퓨터 게임, 증강 현실, 객체 인식, 인간-AI 상호작용, 게임 설계 연구
학번: 2020-27351