Multimodal Technologies and Interaction

MDPI

Article

How New Developers Approach Augmented Reality Development Using Simplified Creation Tools: An Observational Study

Narges Ashtari * and Parmit K. Chilana

School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada; pchilana@sfu.ca
* Correspondence: nashtari@sfu.ca

Abstract: Software developers new to creating Augmented Reality (AR) experiences often gravitate towards simplified development environments, such as 3D game engines. While popular game engines such as Unity and Unreal have evolved to offer extensive support and functionalities for AR creation, many developers still find it difficult to realize their immersive development projects. We ran an observational study with 12 software developers to assess how they approach the initial AR creation processes using a simplified development framework, the information resources they seek, and how their learning experience compares to the more mainstream 2D development. We observed that developers often started by looking for code examples rather than breaking down complex problems, leading to challenges in visualizing the AR experience. They encountered vocabulary issues and found trial-and-error methods ineffective due to a lack of familiarity with 3D environments, physics, and motion. These observations highlight the distinct needs of emerging AR developers and suggest that conventional code reuse strategies in mainstream development may be less effective in AR. We discuss the importance of developing more intuitive training and learning methods to foster diversity in developing interactive systems and support self-taught learners.

Keywords: AR development; software development; information seeking

Citation: Ashtari, N.; Chilana, P.K. How New Developers Approach Augmented Reality Development Using Simplified Creation Tools: An Observational Study. *Multimodal Technol. Interact.* **2024**, *8*, 35. https://doi.org/10.3390/mti8040035

Academic Editor: Arun K. Kulshreshth and Kevin Pfeil

Received: 4 March 2024
Revised: 29 March 2024
Accepted: 9 April 2024
Published: 22 April 2024

1. Introduction

Designing and implementing Augmented Reality (AR) experiences is a complex, knowledge-intensive endeavor that has predominantly been carried out by specialized experts in research labs or professional game development studios. Unlike *mainstream* software development for desktop or web environments where the focus is on flat graphical interfaces and standard input methods, AR developers are tasked with overlaying digital content and experiences onto real-world environments via mobile applications or specialized Head-Mounted Displays (HMDs) like the Apple vision pro. Developers usually have to navigate an intricate web of development frameworks and hardware options to construct three-dimensional (3D) interactions and heighten the realism of their projects [1].

When beginning AR application development, many newcomers gravitate towards traditional 3D game engines, such as *Unity* or *Unreal*, that have evolved to offer extensive support and functionalities specifically for AR development [2]. Unity, for example, streamlines the integration of diverse AR platforms, such as ARKit for iOS and ARCore for Android, into a single, unified Application Programming Interface (API). This integration enables the creation of AR applications that are compatible across various devices and platforms without necessitating platform-specific coding. Furthermore, developers can utilize additional built-in features and resources to expedite the development process and engage with an expansive community of developers [3–5].

Despite the appealing and simplified AR development environments presented by modern game engines, many new creators still find it challenging to realize their immersive development projects [1,6]. To understand the types of obstacles developers face in AR

development, even with simplified environments, a deeper insight into how new AR developers approach programming and debugging is essential. This need has become more pressing with the advent of technologies like the Apple Vision Pro, which aims to bring hundreds of AR applications, from productivity to entertainment [7], to the mass market with its mixed-reality capabilities. It is crucial to support new developers and equip them with the necessary skills to navigate the complexities of AR development. This support will empower new developers to create diverse and innovative immersive applications that fully leverage the potential of advanced AR technologies.

In this paper, we investigate how newcomers in AR approach the creation process using a simplified development environment and seek information to support their design and programming needs. We carried out detailed in-lab task-based observations and semi-structured interviews with 12 software developers who were implementing AR for the first time using the Unity development environment. This choice was made, as prior research [1,8] indicates that while new AR creators have access to a range of development tools not requiring coding skills, they inevitably turn to tools that do require coding due to the flexibility and wide range of functions that allow developers to create comprehensive experiences. By focusing on participants who already have some programming and relevant information-seeking skills, this study directly explores how their foundational knowledge affects their learning curve and adaptation strategies in immersive AR development.

Among our key findings, we found that new AR developers often relied on their previous 2D development experience and sought guidance from online code examples and tutorials. However, these developers faced challenges in applying their 2D experience to the 3D realm. Their usual sources of information, such as online forums and YouTube, were too general and failed to address the unique challenges of AR, including the prediction of 3D object behavior and complex physics. Faced with increasing complexity in AR development, many developers turned to AI-based assistance, only to find that this approach often led to inconsistent results. A primary issue was the developers' narrow focus on finding code snippets, which caused them to overlook the challenges of 3D spatial interactions and the intricacies of AR hardware and software. Additionally, the tendency of developers to dive into coding without a comprehensive understanding of the broader problem and its components proved ineffective in AR development.

Our paper highlights the shortcomings of popular online learning resources and approaches in preparing new developers for the unique challenges of building interactive AR applications. While there is a long history of empirical research exploring developers' work habits in various engineering tasks [9], their learning strategies [10], and online information-seeking behaviors [11,12], our study adds new insights about how software developers tackle interactive immersive experiences and how they navigate complex programming structures and frameworks, and tackle debugging and testing tasks. The lessons learned from our work can be used to invent tailored and more effective learning tools and training programs that empower new creators to explore their own projects in AR. The main contributions of our work are as follows:

1. Providing detailed insights through observations and interviews into how developers new to AR make use of a simplified AR development environment, including their use of various online information resources and AI-assisted tools;
2. Synthesizing the common challenges encountered during AR development, especially related to navigating the unfamiliar intricacies of 3D environments and identifying gaps in their coping methods to tackle these challenges;
3. Identifying opportunities and implications for the design of learning tools and approaches to support future authors of AR applications and help them make a smoother transition from mainstream development.

2. Related Work

This research builds upon insights from Human–Computer Interaction (HCI) and software engineering reflecting on the current landscape of AR tool development, challenges of building domain-specific software, and software developers' information-seeking activities.

2.1. Tool Innovations Is AR Application Development

Prior work has explored various AR-specific authoring tools tailored to creators with diverse skill levels and different fidelity stages of the resulting artifacts [8,13]. Notable examples of such tools include Pronto [14], ProtoAR [15], GestureWiz [16], iaTAR [17,18], ARVIKA [19], Adobe Aero [20], Microsoft Maquette [21], and Reality Composer [22]. These tools have significantly contributed to the field by focusing primarily on supporting the low-to-medium fidelity prototyping stages of application development. Some of these tools strive to minimize or altogether eliminate the need for extensive programming skills and reach a wider creator audiences.

On the other hand, the utility of simplified authoring tools is often limited, as they are overly tailored to predefined tasks, restricting their adaptability to a wide array of platforms, frameworks, and hardware configurations [1,8,23]. Secondly, some of these tools are not universally accessible to end users, either due to their limited availability (beyond research labs), lack of community, or the absence of comprehensive features. Moreover, a significant drawback lies in the fact that these tools seldom cover the entire design cycle, from initial prototyping to subsequent development and testing on AR devices, leaving a critical gap in the seamless progression of the development process as demonstrated in prior work [1].

In practice, commercial AR/VR game engines and software development kits, such as Unity [24], Unreal [25], ARKit [26], ARCore [27], A-Frame [28], and WebXR [29] have emerged as the go-to choices for professionals and enthusiasts alike [1,8,23]. Such tools have stood out due to their robust features, providing extensive documentation, tutorials, and a supportive community for developers. Due to the widespread use of the Unity development platform [30], we studied AR developers' use of Unity to create their first AR application. Our study provides valuable insights into developers' challenges and strategies as they start their first AR development project, complementing existing research [1,6,8,23], offering a comprehensive analysis of real-world AR development practices.

2.2. Domain-Specific Software Development

In this paper, we present an observational study of developers new to the domain of AR. The challenges inherent in domain-specific software development have been well documented, highlighting the varying needs of different user groups across different stages of design [31,32]. For example, prior work has looked at artists using creative coding languages [33,34]. This research has identified the challenges artists face in understanding abstract representations and adapting to structured workflows [35], as well as efforts to support them through platforms tailored to domain-specific requirements [34,36]. The discrepancy between general software engineering practices and their application in scientific programming is also noteworthy, highlighting the need for domain-specific methods and tools tailored to scientists' needs [37–39]. Additionally, game development showcases the unique aspects of specialized domains. It reveals significant differences from other creative industries [40–43] and poses challenges to traditional software development models' predefined phases [44].

Studies in domain-specific challenges of building interactive applications have focused on managing the volatility of these environments, characterized by frequent changes in users, devices, and software components [45–47]. This necessitates systems that can adapt or degrade gracefully amidst changes and failures. The trend towards practical, educational, and assistive technologies mirrors the need to address challenges arising from the dynamic 3D space, affecting user interactions and application requirements. In the realm of AR/VR, prior work [1] provides insights into creators' attitudes and preferences by focusing on a

broader creator population. This research complements the existing works by focusing on observing new software developers in a lab environment and providing insights into the practical challenges of building AR applications. It reveals nuanced aspects of developers' experiences transitioning from 2D to 3D environments, their information-seeking patterns, and use of Generative AI tools such as ChatGPT as an emerging assistive platform.

2.3. Information Seeking in Software Development Tasks

Prior studies of software developers have shed light on their work habits in writing, changing and debugging software [9], their related cognitive processes [10], and their information behavior and needs [11,12]. Modern software development is known to be intertwined with web search today [48,49], with developers frequently issuing search queries to seek answers about how to use an API, understand code functionalities, and troubleshoot various issues [50,51]. Earlier studies with developers [52] indicated that official documentation is often the first point of reference for developers when learning about a new API, but code examples, peer discussions, and hands-on experimentation with APIs have also ranked high [12].

The process of seeking relevant information presents several challenges for software developers. One key challenge is the "vocabulary problem", a term used to describe the difficulty developers face when there is a mismatch between their understanding of the problem and the language used in official documentation or help resources [53]. This can lead to considerable time being spent on sifting through large and complex sets of documentation, often resulting in the reluctance to consult these resources [54,55]. Another issue comes from the dispersion of relevant information across different sources, complicating the task of gathering all the needed details [52,56,57].

We note that much of the existing research has primarily focused on conventional challenges in back-end development and maintenance tasks, rarely capturing the unique challenges faced by developers working in emerging fields like AR where the focus is on creating a compelling user experience that can augment real-world activities. In these new domains, developers are often tasked with creating interactive experiences that involve user input in novel modalities, often through headsets [1]. Our study complements the existing research on developers by shedding light on how developers new to AR will approach the development process using a simplified development environment and to what extent they are able to transfer their existing mainstream programming skills and information-seeking behaviors in this emerging context.

3. Method

To gain deeper insights into the approaches adopted by software developers new to AR, we employed a qualitative research methodology, including in-lab task-based observations and semi-structured interviews. Our main goal was to understand how newcomers make use of a simplified AR development environment, how they seek information to support their design and programming needs, and how their practices compare to mainstream programming tasks.

3.1. Participants and Recruitment

Since we wanted to observe the AR development process using a simplified creation framework and our tasks required programming, we focused on recruiting participants who had training in software development but had not worked on any AR or VR projects in the past. We employed multiple recruitment strategies, including advertising posters at local educational organizations, leveraging personal connections in industry, and utilizing snowball sampling techniques. By adopting these approaches, we aimed to ensure a diverse participant pool in terms of their backgrounds and skills in programming and design. Our recruitment efforts resulted in a total of 12 participants, with a mix of genders (5F/7M), each bringing unique backgrounds and software development roles to the study as summarized in Table 1. All participants indicated they had completed introductory courses in 3D geom-

etry and linear algebra, either at the high school or university level. Participants ranged from having 2–10 years of experience in programming using programming languages such as Java, JavaScript, Python, HTML, C#, and C++. Only two participants (P2 and P9) had brief experience working with Unity for 2D game creation.

Table 1. Participants' demographic information, years of programming experience, role, and programming language proficiencies. All references to "CS" in this table pertain to Computer Science.

ID	Gender (Age)	# Years of Experience and Role	Programming Languages
P1	F (25–34)	7–10, Researcher (CS)	Python, C, Java, R, MATLAB
P2	M (18–24)	4–6, Software engineer	Python, C, Java, C#
P3	M (18–24)	1–3, Student (CS)	C, Java, Python, JavaScript, HTML
P4	M (18–24)	4–6, Student (CS)	C++, C, Java, Python, JavaScript
P5	M (25–34)	10+, Software engineer	C, C++, Python, HTML, JavaScript
P6	F (25–34)	4–6, Researcher (CS)	Python, C++
P7	F (25–34)	4–6, Researcher (CS)	Java, Python, C++, JavaScript
P8	F (25–34)	1–3, Researcher (CS)	Python, HTML
P9	M (18–24)	1–3, Student (CS)	JavaScript, C#, Python
P10	M (25–34)	4–6, Software engineer	C++, Java, JavaScript, Python
P11	F (25–34)	4–6, Software engineer	Python, HTML, JavaScript, C++
P12	M (25–34)	7–10, Software engineer	Python, C++, C#

3.2. In-Lab Observations and Task Design

Choice of Platform: While platforms like A-Frame and Unreal offer similar functionalities for AR development, we opted for Unity as a representative AR development platform in our investigation. We selected Unity as a case-in-point due to its widespread use (with more than 60% of AR/VR content being made by this platform [30]) and rich feature set, including an integrated physics engine that significantly reduces the need for developers to delve into complex physics and mathematics or write intricate code.

Task Selection and Refinement: In constructing the task for this study, we wanted to ensure that it was doable and captured a range of competencies and complexities in creating an interactive AR experience. Our goal was to assess how AR newcomers would approach the development problem and seek relevant information when using a simplified development framework. We consulted industry experts, ran three pilot studies with participants sharing the same characteristics as our main study target group, and iterated on various task configurations to gauge their feasibility.

The task was multi-layered, with each component tailored to test the different skill sets necessary in AR development. We provided participants with all of the required 3D assets (see Figure 1). We instructed participants to create an AR model of the Earth with a continuous spinning motion around its own axis. This component was intended to assess the individuals' capability to introduce and manage elementary motion dynamics in an AR environment. In addition to the spinning Earth, participants were required to incorporate a moon that would not only revolve around the Earth but also execute a spin around its own axis. This layer of complexity ensured that participants dealt with coordinating multiple synchronized motions within the AR space. In the next part of the task, we provided a 3D model of a spaceship and asked participants to simulate a landing mission which required a realistic, physics-driven animation sequence in the AR environment. This involved common AR tasks, including collision detection and defining movement trajectories in a 3D environment. Lastly, the task asked for a simple menu embedded within the AR experience that would have two interactive buttons: "Land" and "Fly Away". Activation of the "Land" button would command the spaceship to initiate a landing sequence onto the moon's surface, while the "Fly Away" button would instigate the spaceship's departure. The inclusion of this component aimed to probe the participants' ability to combine interactivity with immersive visualization, a key proficiency in AR development.

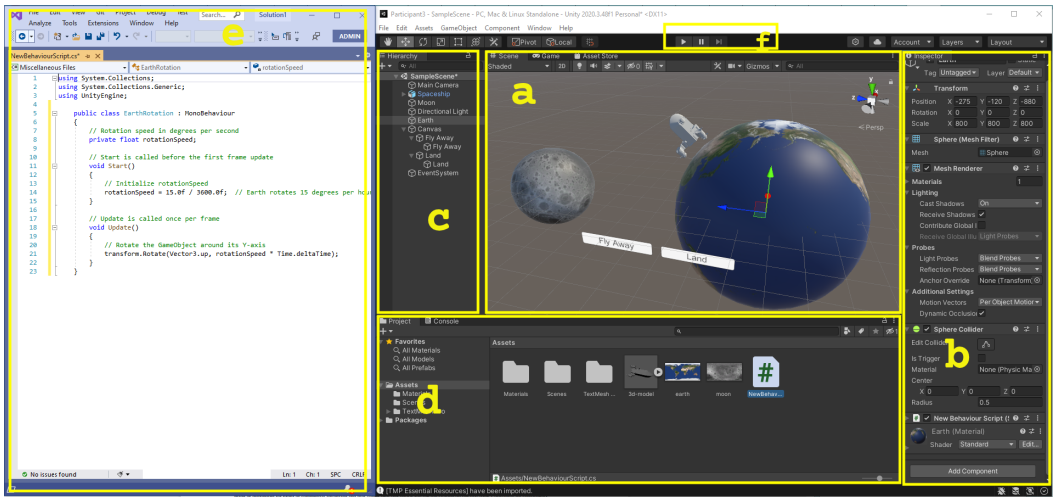

Figure 1. Schematic of the multi-layered AR development task for participants in Unity. (**a**) The "Scene View" where participants interacted with the provided 3D assets. This platform allowed participants to navigate and manipulate 3D objects. (**b**) The "Inspector Panel" which provides properties of the selected 3D object(s) in the scene view. Through this panel, participants were able to modify objects' attributes like position, rotation, scale, and attach components or scripts. (**c**) The "Hierarchy Panel" which lists all the objects in the current scene and provides an easy way to select, organize, and manage game objects. (**d**) The "Project Panel" which is essentially the file browser within Unity and shows all assets, scripts, prefabs, scenes, etc. (**e**) The "Visual Studio" integrated IDE often used in conjunction with Unity for scripting and code editing. (**f**) The "Game View" where participants could preview what they built as it would appear when running. Using this participants were able to play, pause, and step through frames for testing.

Each session lasted around 2 hours, with participants having 90 minutes to complete the task. Participants were informed that they could proceed as far as they could within the 90 min task time frame, and there was no requirement to complete the entire task. Furthermore, the order of implementing different parts of the task was entirely up to the participants; they were free to choose the components they felt most comfortable with or interested in. All participants were instructed to take a break as needed. The study facilitator was readily available throughout the research, offering occasional hints to participants in the event of significant delays in their progress. The facilitator remained discreet to minimize any potential impact on the study outcomes while providing support.

Procedure: Prior to the study (at least five days before the study session), participants were sent comprehensive tutorials on how to use Unity, including guidelines for testing their creations, adding interactions, and navigating the interface. They were granted access to Unity along with its built-in help features and encouraged to utilize any web resources available. To capture an in-depth view of participants' actions and thoughts, their audio and screens were recorded, along with their browser histories, ChatGPT conversations if used (version 4 was set as a default option for all participants), and interactions within Unity UI. Before starting the study tasks, participants completed a questionnaire (for specific questions please refer to Appendix A.1) covering demographic details, educational background, proficiency in various programming languages, and preferred resources for seeking help and information. Participants then were encouraged to follow a "think-aloud" protocol to keep the facilitator updated on their logic, creative process, and any challenges encountered. Scheduled breaks were also provided to ensure participants remained focused and comfortable throughout the study. The facilitator posed questions mid-task to gain additional insights as the activity progressed (for specific questions please refer to Appendix A.2).

For a more nuanced understanding of AR deployment, participants were encouraged to deploy their AR models on the HoloLens 2. This step was critical for assessing the

robustness of their AR experiences in a real-world, immersive setting. To streamline this process and focus on the core objectives of the study, the actual deployment task was handled by the research team, allowing participants to concentrate on conceptualizing and building their AR models.

Post-Task Questionnaire: After completing the task, participants were asked to fill out a short survey questionnaire. The questions aimed to gauge the perceived difficulty of programming in a 3D environment as compared to mainstream, non-3D coding, the effectiveness of online resources in participants' information-seeking process, ease of transferring current programming skills to 3D and AR development tasks, and the extent to which participants relied on their existing programming skills to troubleshoot and solve technical problems while completing the study tasks (for specific questions please refer to Appendix A.3).

3.3. Follow-Up Semi-Structured Interviews

To reflect on the in-lab experiences and better gauge participants' perspectives on AR development, we carried out follow-up semi-structured interviews. Acknowledging the participants' prior experiences in other domains of development, particularly 2D or other non-3D environments, was a crucial component of the post-task interview process. This served to draw comparisons and contrasts, aiming to understand how the unique complexities of AR development diverge from or align with other forms of mainstream software development. One key area that the interview focused on was whether and how the participants had to adapt their existing skills and strategies to the nuances required by AR development. In particular, the semi-structured interviews explored the following:

- **Participants' Experience and Skill Transferability in AR vs. 2D/non-3D Environments:** This focuses on how participants' previous experiences in other environments translate to the AR task. It also aims to identify skills that are easily transferable and those that require significant adaptation or relearning.
- **Challenges, Strategies, and Information-Seeking Behavior:** This combines the specific challenges encountered with the resources and strategies employed to overcome them. It can explore any changes in participants' go-to platforms for assistance and how effective these are, providing insights into their evolving problem-solving process.
- **Lessons Learned and Future Approaches:** This captures personal reflections on what participants would do differently in future similar AR tasks, revealing data on the learning curve involved in AR development.

Each interview concluded with an opportunity for participants to share additional thoughts, feedback, or reflections not covered by the preceding structured questions (for specific questions please refer to Appendix A.4).

3.4. Data Analysis

To analyze the participants' progress, we segmented our primary task into four distinct sub-tasks (as explained in Section 3.2), each revolving around crucial interaction components. Participants' efforts were then aligned with the reference design's sub-tasks. To gain insight into how newcomers identify and utilize various help resources, we initially examined the various phases of their information-seeking behavior in the lab. Additionally, we explored how participants perceived these help resources, drawing on their in-lab interactions while completing the task.

Analysis of Help-Seeking Phases. For our lab-based analysis, we adapted and revised an existing theoretical model on in-person help-seeking by Nelson-Le Gall [58]. Within this framework, we categorized help-seeking behaviors into three main phases: (1) identifying resources; (2) assessing resource relevance; and (3) implementing the relevant assistance to accomplish a task.

Identifying Resources. In this initial phase, we evaluated how effectively participants could articulate their need for help and locate relevant resources. Our assessment tools included a query log analysis of search histories, Unity's built-in help, and an evaluation

of participants' engagement with Generative AI platforms, such as ChatGPT (where applicable). We sought to identify both relevant and irrelevant resources that participants discovered. Additionally, we gauged the time required for participants to initiate their first attempt at seeking help, examining their navigation strategies and the initial moments they initiated assistance using any available resources.

Assessing Resource Relevance. During this phase, we studied what transpired when participants arrived at a potentially relevant resource, as well as how well they leveraged that resource to accomplish their task. For this part of our analysis, one researcher cross-referenced data from participants' browser navigation histories and screen recordings. This helped to evaluate the relevance of the discovered resources and was further substantiated by participants' think-aloud reasoning data during the study.

Implementing Help in Task Completion. In the final phase, we examined the degree to which participants could apply the located help to their current tasks. For this, we relied on browser navigation histories, screen recordings, and participants' attempts at task execution.

Understanding Participants' Perceptions of Help Resources. To delve deeper into participants' perceptions regarding the usefulness of the various help resources they encountered, we drew upon our observations and participants' feedback from post-task questionnaires and semi-structured interviews. Using an inductive analysis method [59], we searched for recurring patterns and themes in the collected data.

4. Results

We present our results by first summarizing the participants' overall performance and approaches to developing an AR experience for the first time when using a simplified development platform, revealing a preference for direct implementation over problem solving. We next present a key analysis of participants' information-seeking activities, highlighting their search queries, preferences for information resources and formats, and the use of Generative AI tools such as ChatGPT. We then delve deeper into the challenges participants faced, which ranged from initial setup difficulties to the intricacies of 3D development and the challenges of translating 2D development knowledge to 3D environments.

4.1. Overview of Task Completion

Our analysis showed that most of our participants faced difficulty in tackling their first AR task: on average, our participants managed to complete a mere 35.8% of the primary task (25% min–75% max) with 34% accuracy (12.5% min–62.5% max). The participants indicated that AR development was either much more difficult (8/12) or somewhat more difficult (4/12) compared to mainstream 2D development tasks (see Figure 2a). At the onset of the study, participants had the option to sketch out or strategically plan their development process. However, without exception, all participants chose to dive directly into implementation, expecting to find a designated area within the Unity user interface for writing code (which existed in the Unity UI; see Figure 1). Despite being encouraged to access tutorials about Unity's interface and methods for adding interactions to 3D objects, none of the participants had consulted these resources prior to attending the user study session. To initiate interaction with the provided 3D objects in the Unity environment, 9 out of 12 participants relied on a trial-and-error approach, while the remaining three participants directly searched on Google. On average, it took participants 4.25 min to initiate their first help-seeking attempt. Although all participants were given access to the Hololens 2 headset to test their code, only 2 participants chose to do so; the remainder opted for utilizing Unity's built-in testing environment to demonstrate and test their projects.

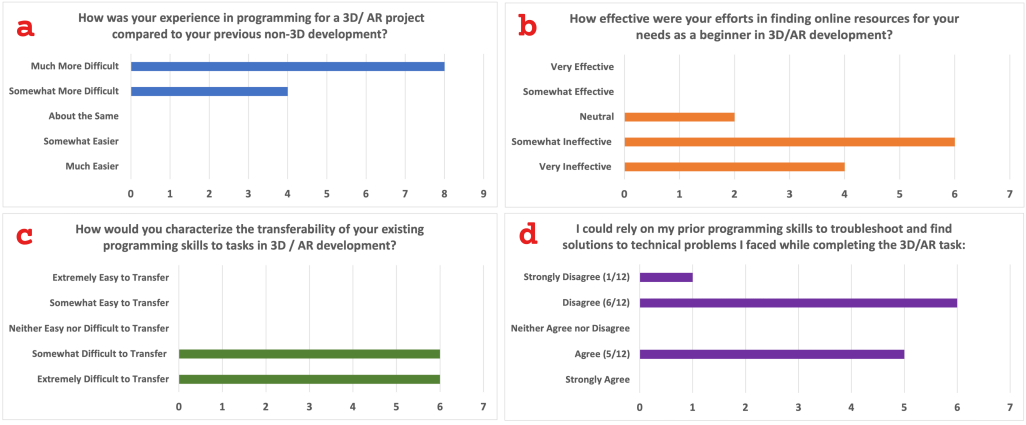

Figure 2. Overview of participants' responses to post-task questionnaire. (**a**) Participants perceived AR development as notably challenging compared to 2D tasks, with most finding it much harder. (**b**) Most participants found their efforts in locating online resources for 3D/AR development ineffective. (**c**) Most participants found it difficult to transfer their existing programming skills to 3D/AR environments. The shift to 3D added complexities in visualizing object movements, affecting not only implementation but also problem-solving approaches. (**d**) Over half of the participants struggled to apply their old "test-and-develop" coding habits in the new 3D/AR environment. For example, they would choose only familiar parts of the code snippets suggested by ChatGPT, but found this strategy unhelpful for debugging intricate AR interactions involving physics forces.

4.2. Overview of Information-Seeking Activities

All participants relied on Google consistently while completing the sub-tasks. Similar to mainstream coding projects, they made use of search results that included official documentation, user forums (e.g., Stack Overflow) and snippets of code found on these platforms, followed by video tutorials (e.g., YouTube). All participants were familiar with ChatGPT during the time of the study and more than half (7/12) ended up using it in the study. On average, participants sought help 13.9 times in each session, with the frequency of these attempts ranging from 11 to 19, a variance of 5.17, and a standard deviation of 2.27 (see Figure 3). A variety of factors triggered these help-seeking activities (discussed below).

Even though we provided pre-study learning resources on the basics of Unity to help onboard participants, most of the participants relied on their own trial-and-error to figure out the Unity interface. This has been commonly seen in other studies of software learning [60], known as the paradox of the active user [61]. Instead of following the Unity documentation, our participants posed queries like "attaching code to 3D objects", "defining objects in Unity 3D", or generic ones like "how to instantiate an object in C#?". This DIY exploration method, while valuable for its hands-on nature, was time-consuming and often directed developers towards unnecessarily intricate solutions. On average, participants toggled 11 times between the coding interface and auxiliary resources, spending 49% (44.5 min) outside the development environment.

Participants' help-seeking behaviors were varied and included searching for learning how to activate specific functionalities and open-source code (55%), integrating multiple or concurrent functions into 3D objects (12%), understanding scaling and metrics (14%), and troubleshooting bugs during code compilation (19%).

In terms of choice of learning strategies and overall task completion rate and accuracy, we saw a range of behaviors among our participants. For example, P1 relied on ChatGPT as the primary learning resource, posing targeted inquiries directly related to the task. This approach facilitated the highest completion rate (75%) of the task with a high level of accuracy among other participants (62.5%). Conversely, P8 dedicated extensive time to reviewing official Unity documentations. However, their ability to apply the broadly acquired knowledge to the specific task was less effective, resulting in minimal progress in task completion (25%). On the other hand, P11 predominantly relied on video tutorials as a

learning resource to complete the tasks. Although they achieved a higher task completion rate (50%) compared to P8, we observed that P11 was mostly trying to recreate the instructions in the video tutorials. As a result, they struggled to implement specific adjustments and had low overall accuracy (25%).

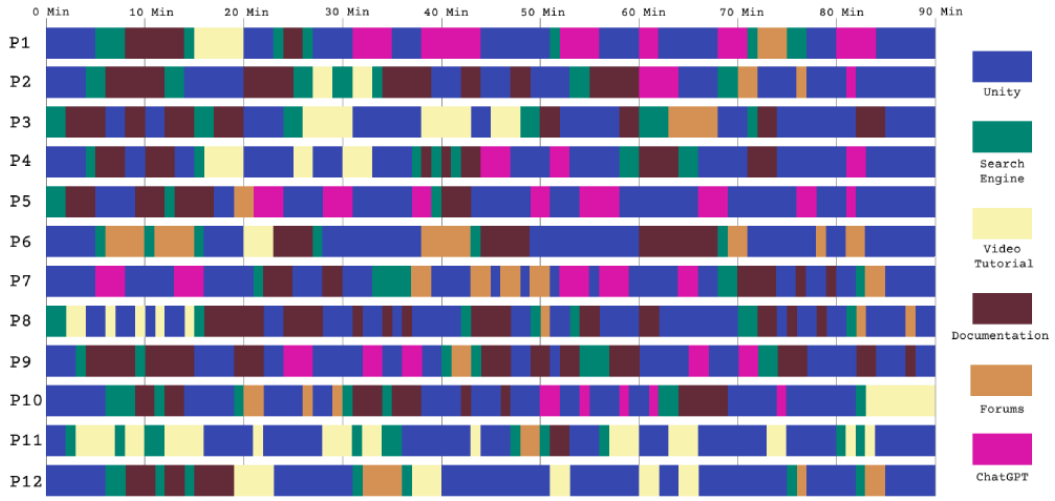

Figure 3. Timeline overview of participants and their information resource and development environment navigation. Participants usually began with Google searches to locate learning resources, often consulting official documentation, forums, and code snippets, with an average of 13.9 queries per session. Despite having access to the basics of Unity prior to starting the study, most participants favored learning by trial-and-error, frequently querying Unity interface specifics. On average, they switched 11 times between coding and resources, spending 44.5 min outside Unity. While initially favoring written documentation for quick lookup, 7 out of 12 participants eventually consulted ChatGPT after other resources proved unhelpful. But, only P1 and P5 found ChatGPT useful for streamlining their learning and workflow.

4.2.1. Preferences for Information Resources and Formats

As seen in previous studies of developers [12], our participants initially sought written documentation, valuing its quick accessibility and skim-readability for coding activities. Rooted in their prior experience with programming languages like Python and Java, participants expected text-based platforms (e.g., Stack Overflow and Unity forums) to offer immediate, relevant support. However, both our observations and participants' post-task feedback (see Figure 2b) indicated that these resources often fell short in addressing the specific challenges tied to crafting 3D interactions. Participants commonly described these resources as either too generic or not directly applicable to their unique requirements. For example, P6 explained, "*As my go-to for quick fixes it felt instinctual to turn there [Stack Overflow] when I began tackling the task. But the resources out there just didn't dig deep enough into the specific issues I was trying to solve or I didn't know how to cater them to my own code. It's one thing to find a code snippet that rotates a shape; it's a whole other ball game to adapt that into a moving, interactive 3D environment*". A recurring struggle for participants, as explained by P6, was in understanding the relevance of the found example code snippets, mapping them onto their own codebases, and modifying them to reflect their specific needs.

Video tutorials, on the other hand, were described as being especially useful for visualizing the procedural steps. For example, P2 explained: "*So even though there is no explanation [in videos], you kind of figure out the steps. . . it's not really good for understanding, it's good for doing. The reason I'm jumping into this kind of video is that I've never done anything with AR. But I have 2D Unity experience. That's why I can understand what's going on here, even though I don't really know the 3D stuff*". However, some participants conveyed that these tutorials lacked depth in elucidating the underlying principles. P3 stated "*I don't like learning from videos. Since they are slow. Usually, when you learn via videos on how to create stuff,*

there's more bloat than there is actual content. I like learning direct…I like to jump straight to the documentation. That's how I learn a lot of new stuff in programming".

4.2.2. Use of Novel Generative AI Tools for Information-Seeking

Despite all participants having previous experience with ChatGPT for programming purposes, only one participant used ChatGPT as the starting point for tackling the task; the majority of participants (11/12) searched for online documentation and forums instead. When these methods failed, more than half of the participants (7/12) tried ChatGPT. However, all of these participants had mixed feelings about using ChatGPT to learn AR development and questioned its reliability based on their coding experiences. For instance, P10 mentioned "*After trying the documentation and getting nowhere, I decided to give ChatGPT a shot. But based on my past encounters, I wasn't too hopeful about its accuracy with coding issues*". In contrast, P1 and P5 heavily used ChatGPT, benefiting from a more focused and efficient approach. Specifically, P1 used detailed prompts and follow-up questions to get accurate, context-sensitive advice from ChatGPT (Figure 4). This strategy helped P1 and P5 grasp foundational concepts quickly and integrate advanced features, thereby improving their workflow.

Participants who asked ChatGPT vague or context-free questions faced difficulties getting tailored answers. For instance, P4 received generic advice that did not fit their existing setup, highlighting the importance of better contextualizing the dialog (Figure 5).

User prompt: "Write Unity C# code for a moon object orbiting itself and Earth."
ChatGPT response (summarized): Delivered script and explanation; user unclear on script implementation.
User prompt: "How to attach this script to the moon object?"
ChatGPT response (summarized): Supplied step-by-step guide for script attachment; user proceeds with creation.
User prompt: "Add rotation for Earth and Moon on their axes. Provide script."
ChatGPT response (summarized): Script and logic explanation given; user uncertain about object interactions and parenting.
User prompt: "How should I run this code?"
ChatGPT response (summarized): Instructions for attaching and running the code provided; user unable to see the effects after testing.
User prompt: "On hitting play, everything resets to zero. What's the issue?"
ChatGPT response (summarized): Received Unity's troubleshooting suggestions; encountered a new problem upon trying to resolve.
User prompt: "Main camera position resets in play mode. How to fix?"
ChatGPT response (summarized): Received irrelevant response from Chat GPT; user seeks clarification.
User prompt: "Scene screen camera doesn't match Game screen. Why?"
ChatGPT response (summarized): User applies the help provided and successfully rotates moon and Earth on their axes and the moon around Earth.

Figure 4. Illustration of a part of P1's dialog with ChatGPT—one of the few examples in the study that shows successful use of AI assistance. P1 initiated the conversation by requesting C# code for a moon orbiting Earth. Faced with uncertainties in the script implementation, they followed up to clarify how to attach the script to the moon object. Subsequent questions involved adding axial rotation to Earth and the Moon, resolving issues related to object interactions and parenting. Additional troubleshooting involved resolving camera position resets and discrepancies between scene and game screens. This iterative, context-rich dialog resulted in more tailored guidance from ChatGPT, enabling P1 to successfully implement the desired orbital and rotational behaviors.

User Prompt: "How do I move an object in Unity?"
ChatGPT Response: You can move an object using Unity's animation system. Create an animation clip with the desired movement and play it to move the object.

Figure 5. Illustration of a part of the P4's unsuccessful interaction with ChatGPT: While P4 already had a script-managed movement system, incorporating animation (recommended by ChatGPT) proved challenging. P4's vague prompt led to generic advice, causing conflicting functions to override each other.

Despite receiving complete code snippets from ChatGPT, many participants only implemented selected parts. This selective use reflected a "test-and-develop" approach based on their previous coding habits. They focused on familiar elements, avoiding more

complex features to sidestep potential confusion or complications. For example, P5 used only part of the code for landing a spaceship on a rotating moon and decided to forgo the complex orientation procedures, stating, "I'll start with what I know and build upon it." While participants often relied on their prior experience, most of them strongly disagreed (1/12) or disagreed (6/12) that their prior skills were sufficient for debugging and tackling other technical problems in AR development (see Figure 2d).

4.3. Challenges in Information-Seeking When Developing AR

Navigating the intricacies of AR development while developing in a simplified environment presented our study participants with a series of nuanced challenges, ranging from difficulties in the initial setup and understanding built-in functionalities, to getting used to the complexities of 3D development and struggles with advanced physics in a 3D environment. The transition from mainstream to 3D development demanded not only a conceptual realignment but also adaptations in troubleshooting and help-seeking approaches. While past work [1] highlights some of the approaches used by newcomers for prototyping AR/VR experiences, our study focuses on the specific approaches and challenges encountered with the use of simplified development platforms in the implementation phase. In the subsequent sections, we explore these challenges and uncover the adaptive strategies used by developers with different levels of expertise.

4.3.1. Challenges in Getting Started

A consistent challenge that we observed among participants was difficulty in identifying a reliable starting point as they were overwhelmed by the array of possibilities. For example, as P3 explains, "*Knowing where to start was my biggest challenge. That's why I was struggling to figure out what resources to use. Using direct resources like YouTube helped. After learning the basics, such as not needing to render lights individually, it became easier to build on my existing programming skills*".

Strategies to "*cope with the unknowns*" varied across the board, echoing distinct developer personas as outlined in Clarke's 2007 study [62]. For instance, P1, who closely resembled the *Pragmatic Developer*, tackled the complex task by breaking it down into smaller, more manageable sub-tasks. The segmented approach improved progress and morale but had drawbacks, including frequent context-switching and difficulty in grasping the overall objectives and best use of learning materials like code examples.

This challenge of achieving a holistic understanding was a broader issue that we observed in the study. A trend among the majority of participants (10/12) was the use of a granular, bottom–up strategy—breaking down the task into basic queries like "how to move an object in Unity" without understanding the overall development task.

On the flip side, two out of the twelve participants managed to substantially complete their tasks by initially adopting a bottom–up strategy and then transitioning to a top–down approach, exhibiting traits of the *Opportunistic Developer*. In particular, P1 and P5 consulted documentations and YouTube tutorials to master foundational concepts before moving to structure their code more comprehensively. Then, they leveraged ChatGPT to benefit from its dialogic interaction, tailored advice and context-specific solutions, thereby obviating the need to "reinvent the wheel". This involved turning to platforms like ChatGPT and YouTube to understand the relationships between different coding components.

4.3.2. Lack of Awareness of Development Framework's Built-In Interactions and Affordances

Participants faced challenges in distinguishing between Unity's built-in functions and custom code when looking for help in online documentation, often leading to unnecessary debugging time. For instance, P12 said, "*I saw Transform.Translate and thought I needed to define it myself. Big mistake. It conflicted with Unity's built-in method and slowed my progress*". Similarly, P8 added, "*When I looked at the code samples in Unity forums, they mentioned 'Quaternion' for orientation and 'Rigidbody' for physics interactions. I couldn't tell if these were*

built-in features or if they were custom-defined in the script. It took me a while to figure out that these are standard Unity components, and I didn't have to define them myself".

In a related observation, P7 spent significant time manually scripting the spaceship's movement, unaware that Unity's built-in functions could simplify the task. The participant's searches like "object direction in Unity 3D" did not lead to discovering Unity's built-in functionality (NavMesh), partly because of vocabulary issues and the advanced language used in Unity's documentation. This made it difficult for participants with non-AR related vocabulary to find efficient solutions or identify specific Unity features in the provided code samples.

4.3.3. Difficulties in Navigating 3D Environments

Overall, all participants found it either somewhat difficult (6/12) or extremely difficult (6/12) to transfer their existing programming skills to 3D/AR development tasks (see Figure 2c). Furthermore, although all participants had completed courses in 3D geometry and linear algebra at the high school or university level, they found it challenging to apply this theoretical knowledge practically in the development of 3D experiences. Their familiarity with mathematical foundations did not necessarily translate into the ability to implement these principles in real-world development scenarios. Even with the availability of built-in physics capabilities and testing features in Unity, participants faced difficulties in visualizing object movements in 3D space. This change in complexity influenced not only their interaction but also their methods of seeking help. Supporting this point, P11 mentioned "*In 2D environments, I could sketch out motions and interactions on a piece of paper. . . like reading a map. But now it seems like shifting from a map to a globe. Suddenly, you're accounting for depth, orientation, and multi-dimensional interactions. . . I'm not sure even if I can ask the right question or if my search prompts would work*".

Furthermore, the intricacies of 3D visualization were not the only aspect that perplexed participants; the shift from the right-handed coordinate system commonly found in mainstream 2D coding environments to Unity's left-handed system was particularly disorienting. Participants attributed this confusion to their previous experience with standard screen-based 2D graphics (e.g., 2D game development, web development, and HTML5). Misalignments and unexpected behaviors due to this shift were evident. As P4 noted, "Transitioning from my usual coding environment [web development] to Unity's left-handed coordinate system was quite disorienting. My 3D models seemed completely out of place, like they were twisted and scattered across the scene." In their search for solutions, participants frequently drew on terms from their prior knowledge in 2D environments. Common search queries included "Unity 3D vs. 2D coordinates" and "2D to Unity 3D transition", with the prefix "2D to 3D" being recurrent.

4.3.4. Challenges of Dealing with Multiple Physics Forces and Predicting 3D Object Behavior in AR

Unlike 2D programming, object interactions in 3D environments like AR often involve more sophisticated considerations. For example, we observed that participants faced unexpected behavior while coding a spaceship landing on a rotating sphere, unaware that complex physics forces were at play. As P1 stated, "*If it was common game dev I had my spaceship perfectly landed; Introducing rotation threw everything off not giving a hint where the problem is coming from*".

The unexpected trajectories the participants encountered made them assume coding mistakes, leading them to assume the existence of potential bugs or development platforms inefficiencies. Typical search queries were along the lines of "unexpected physics behavior during landing" and "debugging landing sequence in Unity 3D". This hinted at a potential blind spot in the learning materials: the fundamental physics at play. We observed that despite participants' attempts for finding learning materials, none explicitly mentioned or hinted at the involvement of multiple physics forces in the task at hand. This observation aligns with the findings of Ashtari et al. [1], who highlighted the "physical aspects of

the debugging process that remain neglected in online tutorials" of AR/VR development underlining the often-overlooked yet critical aspects of practical implementation.

In addition, predicting object behavior in a 3D environment was challenging due to the complex variables at play, such as rotation speeds, initial positions, and landing trajectories. To decipher the behavior of the spaceship moving from the Earth to the Moon, P8 opted for console-based methods (common in 2D development), and used the print function to analyze raw data such as force vectors and rotation angles. P8's attempt to apply 2D console-based methods quickly proved problematic: the data overload in the console made it tough to identify specific issues, and the numbers lacked the visual context needed for understanding 3D orientations and trajectories. P8 explained "*In Pygame, I could just throw in print statements and quickly figure out what's going on…the numbers directly translated to on-screen coordinates, making it intuitive…I tried the same print-everything approach and got swamped with numbers*".

P8's conventional method of breaking down and testing the code in parts, typically effective in simpler or 2D environments, did not simplify the complexities of the 3D AR task. Instead, it added to the confusion, making it even more challenging to identify the source of the problem. As a result, P8 resorted to making ill-defined and generic queries queries using phrases "object orientation in 3D" and "understanding object movement in 3D" in search engines that were not specific enough to direct them to relevant learning resources. This mismatch between the complexity of the task and the search queries led to an ineffective cycle of problem solving.

5. Discussion

5.1. Key Takeaways

Our findings contribute insights into how software developers new to AR will approach the development process using a simplified development environment. We also looked at the types of information resources they seek and how their learning experience in AR differs from mainstream software development. In particular, we shed light on the unique needs of emerging AR developers, indicating that the types of code reuse approaches that are successful in other development domains may translate poorly AR due to difficulties with unfamiliar vocabularies and intricacies of 3D environments, physics, and motion. Our observations complement and extend prior works that focus on elective, task-focused learning approaches in mainstream software development [63,64], suggesting that the same "immediate solutions" mindset may be ill suited for the complexities of AR development. Moreover, while emerging Generative AI tools are showing promising gains in development tasks [48,63], our research offers an initial look at how these tools may be inadequate for AR development and further widen the skills gap.

With the advancements in mixed-reality devices such as the release of Apple Vision Pro and the increasing interest in AR technologies, more developers are entering the field of AR development. Many of these developers are learning the necessary skills through online resources and adopting an informal approach to their education. This trend underscores the importance of understanding the challenges these new developers face as they navigate the complexities of AR development without formal training. In light of the findings from our study, it is likely that these new developers will face the same challenges that the participants in our study encountered. We now reflect on the implications of our findings for future research in HCI and the need to reconsider the design of training programs and learning resources to effectively support the growing community of informal AR developers who use simplified development frameworks. We discuss ways to enhance AR development by integrating problem-solving strategies, leveraging in-context personalized approaches and Generative AI tools, and increasing user engagement through adaptive feedback and milestone integration.

5.2. Enhancing AR Development Resources with Problem-Solving Strategies

Our study contributes novel insights into developers' challenges with programming AR for the first time using a simplified development framework. Our results build upon prior research that shows that developers generally prefer selective, task-focused learning and information-seeking tactics. Previous studies have highlighted that developers often seek immediate solutions to specific problems [65,66], focusing on getting particular API functions to work [51] or finding workarounds, rather than gaining a comprehensive understanding of the software or its underlying principles [56]. This approach is partly due to the complexity and time-intensive nature of software comprehension, leading developers to prioritize task completion over in-depth understanding.

Our study also shows that AR development presents unique challenges that extend beyond coding skills, requiring a deep understanding of the interplay between 3D elements and real-world physics. Developers often misattribute AR anomalies, like unpredictable object behavior, to coding errors, overlooking the crucial role of physics. This gap in understanding underscores deficiencies in current educational resources and debugging techniques, which are inadequate for addressing the complex interactions in AR. Our findings suggest an urgent need for specialized educational materials and tools tailored to AR development that can, for example, build on the literature on problem decomposition in computer science [67–69]. These resources may include simulation environments visually representing real-world physics affecting digital elements. By offering such insights, developers can better understand the multifaceted challenges unique to AR, enhancing troubleshooting and deepening foundational understanding for success in AR development.

5.3. Improving Learning through In-Context Personalized Approaches

Standard debugging tools may fall short of understanding and predicting user behavior and application performance. To address this, understanding the task context [70–72] is essential. For example, the incorporation of advanced logging and machine learning techniques such as collaborative filtering [73–75] and content-based filtering [76] into development environments like Unity could be game-changing in enhancing both coding efficiency and conceptual understanding. Collaborative filtering would leverage community contributions and feedback to identify common challenges and propose vetted solutions, while content-based filtering could offer tailored recommendations based on an individual users' coding history and behavior within the platform.

The promise of these machine learning methods could be further enriched by integrating AI tools similar to GitHub Copilot into the development environment. Such tools could simultaneously suggest appropriate code snippets and dynamically link developers to contextual learning resources. For instance, if a developer struggles with object physics and encounters errors, the integrated system could suggest an optimized code snippet tailored for Unity's physics engine and couple it with a targeted tutorial that unpacks the relevant physics principles. In doing so, the developer gains not just a quick fix but also a deeper, foundational understanding of the challenge at hand.

5.4. Leveraging Help-Seeking through Generative AI Platforms

Our study findings also shed light on the role of Generative AI tools, like ChatGPT, in AR development, expanding upon existing research on the impact of AI on programmers. Previous studies have shown that tools like GitHub Copilot [77] are effective in generating foundational code and suggesting structures [63,64], but their effectiveness varies based on the nature of queries and developers' ability to articulate their intent in natural language [78,79]. Similar to prior observations [80], we also found that developers struggled to form a mental model of the underlying Large Language Models (LLMs) and to express their intentions accurately.

We also found that detailed, context-rich questions yield more useful responses from ChatGPT (e.g., by P1 and P5), but there are challenges when users lack sufficient context or the ability to validate responses. Selective code snippet implementation from ChatGPT

underscores participants' cautious "test-and-develop" approach based on prior coding experiences. Initial evidence suggests that Generative AI systems, like ChatGPT, in the AR development context, require reconsideration. Incorporating features like grounded utterances [48], converting vague queries into executable code, and step-by-step guides can address the contextual gap, providing educational tools for understanding the "what", "why", and "how" of coding problems. Allowing multimodal inputs such as images or videos could enhance user intent clarity, making AI more effective for those lacking specialized vocabulary. Future versions of Generative AI with these features could offer targeted, adaptable, and educational support for programming tasks. Overall, our study highlights the importance of understanding the strengths and limitations of AI tools for effective use and training in domain-specific software development tasks.

5.5. Enhancing User Engagement and Learning through Adaptive Feedback and Milestone Integration

In the design of systems from an HCI perspective, system feedback and milestones are critical for user engagement and task completion [81]. Our study's findings underscore the importance of creating a progression model that not only facilitates the breaking down of complex tasks into manageable sub-tasks but also ensures that these micro-goals contribute to a holistic understanding of larger objectives of AR development. Our study indicates that breaking complex tasks into smaller, manageable sub-tasks is beneficial, particularly for the "Pragmatic Developer" persona. However, this approach has a downside: while it boosts morale through incremental success, it can hinder a holistic understanding of overarching objectives. The challenge for HCI designers is to strike a balance between immediate micro-level feedback and broader macro-level insights that align with long-term goals, perhaps through a progress tracking dashboard that connects the dots between these two levels.

Furthermore, there is potential to design milestones that are both adaptive, catering to different user personas, and instructive, guiding users toward larger learning goals as they tackle more complex development tasks in AR. The challenge extends to helping users switch between bottom–up and top–down approaches when they hit learning challenges. HCI researchers could focus on developing systems capable of recognizing such learning gaps and offering guidance to navigate through them, thereby ensuring a more rounded educational experience.

5.6. Limitations

Although newcomers to AR can include a range of creators, such as hobbyists and domain experts with varying levels of programming expertise as demonstrated in prior work [1], in this study, we focused on trained developers, as our tasks required programming knowledge. Future work should consider end-user programmers who are not only new to AR but also new to programming, to obtain a different perspective on their challenges. Furthermore, our method relied on in-lab task-based observations and interviews, as this was an appropriate way to capture the emergent nature of AR development and identify key development issues in this rapidly evolving field. But this may not capture the full range of experiences, challenges, and adaptive mechanisms that AR developers encounter in real-world settings. Longitudinal studies involving real-world tasks and larger projects could provide a more comprehensive view. Future work can also incorporate a control group of experienced AR developers for a more rigorous analysis. The current approach was taken because it offers a focused look at the challenges newcomers face, given that there is a rising tide of them entering the field, and existing literature has not yet sufficiently investigated this demographic. This focus provides foundational insights for educational initiatives aimed at supporting this growing community. Lastly, in this study, we concentrated exclusively on AR development to ensure data consistency and methodological rigor; however, it is imperative for future research to broaden the scope to include other extended reality (XR) modalities like Virtual Reality (VR). This will provide

a more nuanced and in-depth understanding of the challenges and learning processes associated with designing and developing activities in 3D environments.

6. Conclusions

In conclusion, our study demonstrates that software developers new to AR struggle in creating interactive immersive AR experiences, even when they are working within simplified development environments. They tend to rely heavily on mainstream information resources during the development process, but these resources are usually inadequate for navigating the intricacies of the AR spatial concepts and physics, and the hardware–software interplay inherent in AR development. The conventional information-seeking strategies approaches fall short in addressing the complex 3D nature of AR. These findings highlight a pressing need for specialized AR training programs and online educational resources that focus on AR-specific problem decomposition rather than optimizing for code-level assistance. By incorporating diverse perspectives and experiences, training programs and resources can be designed to be more inclusive, thereby enriching the AR development ecosystem with a wider range of creative solutions and applications. By shedding light on the specific hurdles faced by newcomers, our work serves as a foundational step towards the creation of more targeted, user-centered learning aids that can better bridge the widening skills gap.

Author Contributions: Conceptualization, N.A. and P.K.C.; methodology, N.A. and P.K.C.; validation, N.A. and P.K.C.; formal analysis, N.A.; investigation, N.A.; writing—original draft preparation, N.A.; writing—review and editing, N.A. and P.K.C.; visualization, N.A.; supervision, P.K.C.; funding acquisition, P.K.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Natural Science Engineering Research Council of Canada grant number P-RGPIN-5444.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and approved by the Ethics Board of Simon Fraser University (Study Number 20190145, 3 February 2023).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study to publish this paper.

Data Availability Statement: The data presented in this article cannot be shared publicly because of ethical considerations.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A

Appendix A.1. Pre-Study Questionnaire

1. Select your age group:
   - 18–24
   - 25–34
   - 35–44
   - 55–65
   - >65
2. What is your current position? (Please do not specify where you work or study.)
   (Sample answer: Student, software developer, math teacher)
3. How many years of experience do you have in programming?
   - 0
   - 1–3
   - 4–6

7–10
More than 10

4. List all of the programming languages you are familiar with, in order of experience/familiarity (List the most familiar first).
5. Do you have any experience working with any AR/VR development frameworks? If so, please explain briefly.

Appendix A.2. Mid-Task Interview Questions

1. How have you progressed in your task so far?

   (Please describe your progress, including any milestones reached or objectives completed).
2. What challenges did you face, if any?

   (Detail any obstacles encountered and how they impacted your work).
3. What is your strategy for the rest of the session? For example, about the approach you are going to take for the rest of your development process and any particular learning resources you want to use or keep using?

   (Outline your plan for moving forward, including any changes to your methodology or resources).
4. Describe a specific feature or functionality that you implemented so far in creating the AR experience. What was your thought process behind it? Please explain the steps you have taken so far to achieve it.

Appendix A.3. Post-Task Questionnaire

1. How was your experience in programming for a 3D and immersive experience compared to your previous non-3D development?

   Much more difficult
   Somewhat more difficult
   About the same
   Somewhat easier
   Much easier
2. How effective were your efforts in finding online resources for your needs as a beginner in 3D/AR development?

   Very effective
   Somewhat effective
   Neutral
   Somewhat ineffective
   Very ineffective
3. As a person with prior experience in programming, how would you characterize the transferability of your existing programming knowledge into creating an immersive 3D/AR experiences?

   Extremely easy to transfer
   Somewhat easy to transfer
   Neither easy nor difficult to transfer
   Somewhat difficult to transfer
   Extremely difficult to transfer
4. As a person with prior experience in programming, I relied on my programming skills to troubleshoot and find solutions to technical problems…

   Strongly disagree
   Disagree
   Neither agree nor disagree
   Agree

Strongly agree

Appendix A.4. Post-Task Interview Questions

1. Show me what you created in today's session.
   (a) How was your overall experience in completing the task?
   (b) What do you think went well today as you were creating your first AR app?
   (c) How did you plan and organize the development process of the AR/VR creation task?
2. Can you tell me about the most challenging parts of this task and any roadblocks you faced while completing your task?
3. What types of learning resources did you use during the creation process (if any)? For example, tutorials, videos, documentation, etc.
4. To what extent were you satisfied with the resources you used when creating your AR/VR application?
5. How did you approach the use of resources you previously mentioned in your development process? Were there any occasions in your development process in which you might have preferred using a help resource over the other options? Please explain.
6. What types of issues or bugs did you encounter during the development of interactive elements of the AR experience (if any)? How did you go about identifying and fixing them?
7. Overall, how does your experience in creating an AR experience for the first time compare to other kinds of software development that you may have done in the past?
8. While completing your task, did you find any capabilities missing from the tool you were using that might have helped you in doing your task better?
9. Reflecting on your experience, what would you do differently if you had the opportunity to start the AR development process again?
10. Is there anything else you would like to share about your experience?

References

1. Ashtari, N.; Bunt, A.; McGrenere, J.; Nebeling, M.; Chilana, P.K. Creating Augmented and Virtual Reality Applications: Current Practices, Challenges, and Opportunities. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 25–30 April 2020; pp. 1–13. [CrossRef]
2. Stack Overflow. 2021 Developer Survey. Available online: https://insights.stackoverflow.com/survey/2021 (accessed on 12 February 2024).
3. Stepico. Why Is Unity the Best Game Engine: Pros and Cons. Available online: https://stepico.com/blog/why-is-unity-the-best-game-engine-pros-and-cons/ (accessed on 13 February 2024).
4. TechCrunch. How Unity Built the World's Most Popular Game Engine. Available online: https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/ (accessed on 17 October 2019).
5. Whimsy Games. Godot vs. Unity: All You Need to Know. Available online: https://whimsygames.co/blog/godot-vs-unity-all-you-need-to-know/ (accessed on 13 February 2024).
6. Ashtari, N.; Alamzadeh, P.; Ganapathy, G.; Chilana, P. PONI: A Personalized Onboarding Interface for Getting Inspiration and Learning About AR/VR Creation. In Proceedings of the Nordic Human-Computer Interaction Conference, Aarhus, Denmark, 8–12 October 2022; Association for Computing Machinery: New York, NY, USA, 2022; Article No. 32. [CrossRef]
7. Apple. Apple Announces More Than 600 New Apps Built for Apple Vision Pro. Available online: https://www.apple.com/newsroom/2024/02/apple-announces-more-than-600-new-apps-built-for-apple-vision-pro/ (accessed on 16 February 2024).
8. Nebeling, M.; Speicher, M. The Trouble with Augmented Reality/Virtual Reality Authoring Tools. In Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Munich, Germany, 16–20 October 2018. [CrossRef]
9. LaToza, T.D.; Venolia, G.; DeLine, R. Maintaining Mental Models: A Study of Developer Work Habits. In Proceedings of the 28th International Conference on Software Engineering (ICSE '06), Shanghai, China, 20–28 May 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 492–501. [CrossRef]
10. Fleming, S.D.; Scaffidi, C.; Piorkowski, D.; Burnett, M.; Bellamy, R.; Lawrance, J.; Kwan, I. An Information Foraging Theory Perspective on Tools for Debugging, Refactoring, and Reuse Tasks. *ACM Trans. Softw. Eng. Methodol.* **2013**, *22*, 1–41. [CrossRef]
11. Kim, A.S.; Ko, A.J. A Pedagogical Analysis of Online Coding Tutorials. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, Seattle, WA, USA, 8–11 March 2017; pp. 321–326. [CrossRef]

12. Ko, A.J.; DeLine, R.; Venolia, G. Information Needs in Collocated Software Development Teams. In Proceedings of the 29th International Conference on Software Engineering (ICSE '07), Minneapolis, MN, USA, 20–26 May 2007; pp. 344–353. [CrossRef]
13. Hampshire, A.; Seichter, H.; Grasset, R.; Billinghurst, M. Augmented Reality Authoring: Generic Context from Programmer to Designer. In Proceedings of the 18th Australia Conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments (OZCHI '06), Sydney, Australia, 20–24 November 2006; Association for Computing Machinery, New York, NY, USA, 2006; pp. 409–412. [CrossRef]
14. Leiva, G.; Nguyen, C.; Kazi, R.H.; Asente, P. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20), Honolulu, HI, USA, 25–30 April 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–13. [CrossRef]
15. Nebeling, M.; Nebeling, J.; Yu, A.; Rumble, R. ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18), Montreal, QC, Canada, 21–26 April 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–12. [CrossRef]
16. Speicher, M.; Nebeling, M. GestureWiz: A Human-Powered Gesture Design Environment for User Interface Prototypes. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18), Montreal, QC, Canada, 21–26 April 2018. [CrossRef]
17. Lee, G.A.; Kim, G.J.; Billinghurst, M. Immersive Authoring: What You EXperience Is What You Get (WYXIWYG). *Commun. ACM* **2005**, *48*, 76–81. [CrossRef]
18. Lee, G.A.; Nelles, C.; Billinghurst, M.; Kim, G.J. Immersive Authoring of Tangible Augmented Reality Applications. In Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR '04), Arlington, VA, USA, 5 November 2004; pp. 172–181. [CrossRef]
19. Friedrich, W. ARVIKA-augmented reality for development, production and service. In Proceedings of the International Symposium on Mixed and Augmented Reality, Darmstadt, Germany, 1 October 2002; pp. 3–4. [CrossRef]
20. Adobe. Adobe Aero. Available online: https://www.adobe.com/ca/products/aero.html (accessed on 11 November 2023).
21. Microsoft. Microsoft Maquette. Available online: https://learn.microsoft.com/en-us/windows/mixed-reality/design/maquette (accessed on 11 November 2023).
22. Apple. Reality Composer. Available online: https://developer.apple.com/augmented-reality/tools/ (accessed on 11 November 2023).
23. Krauß, V.; Boden, A.; Oppermann, L.; Reiners, R. Current Practices, Challenges, and Design Implications for Collaborative AR/VR Application Development. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021. [CrossRef]
24. Unity Technologies. Unity Website. Available online: https://unity.com/ (accessed on 11 November 2023).
25. Epic Games. Unreal Engine. Available online: https://www.unrealengine.com/ (accessed on 11 November 2023).
26. Apple. ARKit. Available online: https://developer.apple.com/augmented-reality/ (accessed on 11 November 2023).
27. Google. ARCore. Available online: https://developers.google.com/ar (accessed on 11 November 2023).
28. Marcos, D.; McCurdy, D.; Ngo, K. A-Frame. Available online: https://aframe.io/ (accessed on 11 November 2023).
29. Vukićević, V.; Jones, B.; Smus, B. WebXR. Available online: https://en.wikipedia.org/wiki/WebXR (accessed on 11 November 2023).
30. Unity Technologies. Unity News Board. Available online: https://unity.com/our-company/newsroom/unity-technologies-finds-63-creatives-struggle-implementing-ar-advertising (accessed on 11 November 2023).
31. Chilana, P.K.; Wobbrock, J.O.; Ko, A.J. Understanding Usability Practices in Complex Domains. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, Atlanta, GA, USA, 10–15 April 2010; ACM: New York, NY, USA, 2010. [CrossRef]
32. Gulliksen, J.; Sandblad, B. Domain-specific design of user interfaces. *Int. J. Hum.-Comput. Interact.* **1995**, *7*, 135–151. [CrossRef]
33. Lieberman, Z. Of Book, a Collaboratively Written Book about openFrameworks. 2014. Available online: https://openframeworks.cc/ofBook/chapters/foreword.html (accessed on 15 February 2024).
34. Reas, C.; Fry, B. Processing. Available online: https://processing.org/ (accessed on 11 November 2023).
35. Li, J.; Hashim, S.; Jacobs, J. What We Can Learn from Visual Artists about Software Development. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21), Yokohama, Japan, 8–13 May 2021; ACM: New York, NY, USA, 2021. [CrossRef]
36. Levin, G. Essay for Creative Code. Available online: https://www.flong.com/archive/texts/essays/essay_creative_code/index.html (accessed on 11 November 2023).
37. Heroux, M.A.; Bartlett, R.A.; Howle, V.E.; Hoekstra, R.J.; Hu, J.J.; Kolda, T.G.; Lehoucq, R.B.; Long, K.R.; Pawlowski, R.P.; Phipps, E.T.; et al. An Overview of the Trilinos Project. *ACM Trans. Math. Softw.* **2005**, *31*, 397–423. [CrossRef]
38. Kelly, D. Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *J. Syst. Softw.* **2015**, *109*, 50–61. [CrossRef]
39. Segal, J. When Software Engineers Met Research Scientists: A Case Study. *Empir. Softw. Eng.* **2005**, *10*, 517–536. [CrossRef]
40. Kultima, A.; Alha, K. "Hopefully everything I'm doing has to do with innovation": Games industry professionals on innovation in 2009. In Proceedings of the 2010 2nd International IEEE Consumer Electronics Society's Games Innovations Conference, Hong Kong, China, 21–23 December 2010; pp. 1–8. [CrossRef]

41. Murphy-Hill, E.; Zimmermann, T.; Nagappan, N. Cowboys, Ankle Sprains, and Keepers of Quality: How is Video Game Development Different from Software Development? In Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 31 May–7 June 2014; pp. 1–11, ISBN 9781450327565. [CrossRef]
42. Tschang, T.F. Video games as interactive experiential products and their manner of development. *Int. J. Innov. Manag.* **2005**, *9*, 103–131. [CrossRef]
43. Ampatzoglou, A.; Stamelos, I. Software engineering research for computer games: A systematic review. *Inf. Softw. Technol.* **2010**, *52*, 888–901. [CrossRef]
44. Stacey, P.; Nandhakumar, J. A temporal perspective of the computer game development process. *Inf. Syst. J.* **2009**, *19*, 479–497. [CrossRef]
45. Kindberg, T.; Barton, J.; Morgan, J.; Becker, G.; Caswell, D.; Debaty, P.; Gopal, G.; Frid, M.; Krishnan, V.; Morris, H.; et al. People, places, things: Web presence for the real world. In Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications, Los Alamitos, CA, USA, 7–8 December 2000; pp. 19–28. [CrossRef]
46. Schiele, G.; Handte, M.; Becker, C. Pervasive Computing Middleware. In *Handbook of Ambient Intelligence and Smart Environments*; Springer: Boston, MA, USA, 2010; pp. 201–227. [CrossRef]
47. Abowd, G.D. Software Engineering Issues for Ubiquitous Computing. In Proceedings of the 21st International Conference on Software Engineering, Los Angeles, CA, USA, 16–22 May 1999; pp. 75–84. [CrossRef]
48. Liu, M.X.; Sarkar, A.; Negreanu, C.; Zorn, B.; Williams, J.; Toronto, N.; Gordon, A.D. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, 23–28 April 2023; Article No. 598.
49. Sadowski, C.; Stolee, K.T.; Elbaum, S. How Developers Search for Code: A Case Study. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015), Bergamo, Italy, 30 August 2015; pp. 191–201. [CrossRef]
50. Li, H.; Xing, Z.; Peng, X.; Zhao, W. What help do developers seek, when and how? In Proceedings of the 2013 20th Working Conference on Reverse Engineering (WCRE), Koblenz, Germany, 14–17 October 2013; pp. 142–151. [CrossRef]
51. Sillito, J.; Murphy, G.C.; De Volder, K. Asking and Answering Questions during a Programming Change Task. *IEEE Trans. Softw. Eng.* **2008**, *34*, 434–451. [CrossRef]
52. Robillard, M.P. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Softw.* **2009**, *26*, 27–34. [CrossRef]
53. Furnas, G.W.; Landauer, T.K.; Gomez, L.M.; Dumais, S. The vocabulary problem in human-system communication. *Commun. ACM* **1987**, *30*, 964–971. [CrossRef]
54. Carroll, J.M. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*; MIT Press: Cambridge, MA, USA, 1990.
55. Novick, D.G.; Andrade, O.D.; Bean, N. The Micro-Structure of Use of Help. In Proceedings of the 27th ACM International Conference on Design of Communication, Bloomington, IN, USA, 5–7 October 2009; pp. 97–104. [CrossRef]
56. Maalej, W.; Tiarks, R.; Roehm, T.; Koschke, R. On the Comprehension of Program Comprehension. *ACM Trans. Softw. Eng. Methodol.* **2014**, *23*, 31. [CrossRef]
57. Robillard, M.P.; Deline, R. A Field Study of API Learning Obstacles. *Empir. Softw. Eng.* **2011**, *16*, 703–732. [CrossRef]
58. Nelson-Le Gall, S. Chapter 2: Help-Seeking Behavior in Learning. *Rev. Res. Educ.* **1985**, *12*, 55–90. [CrossRef]
59. Corbin, J.; Strauss, A. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*; SAGE Publications: Thousand Oaks, CA, USA, 2014; ISBN 9781483315683. Available online: https://books.google.ca/books?id=hZ6kBQAAQBAJ (accessed on 15 February 2024).
60. Kiani, K.; Cui, G.; Bunt, A.; McGrenere, J.; Chilana, P.K. Beyond "One-Size-Fits-All": Understanding the Diversity in How Software Newcomers Discover and Make Use of Help Resources. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, UK, 4–9 May 2019; pp. 1–14. [CrossRef]
61. Carroll, J.M. *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*; The MIT Press: Cambridge, MA, USA, 1987.
62. Clarke, S.J. What is an End User Software Engineer? End-User Software Engineering. 2007. Available online: https://api.semanticscholar.org/CorpusID:28220424 (accessed on 15 February 2024).
63. Barke, S.; James, M.B.; Polikarpova, N. Grounded Copilot: How Programmers Interact with Code-Generating Models. *arXiv* **2022**, [CrossRef]
64. Vaithilingam, P.; Zhang, T.; Glassman, E.L. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In Proceedings of the Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April–5 May 2022; Article No. 332. [CrossRef]
65. Brandt, J.; Guo, P.J.; Lewenstein, J.; Dontcheva, M.; Klemmer, S.R. Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA, 4–9 April 2009; pp. 1589–1598. [CrossRef]
66. Hou, D.; Li, L. Obstacles in Using Frameworks and APIs: An Exploratory Study of Programmers' Newsgroup Discussions. In Proceedings of the 2011 IEEE 19th International Conference on Program Comprehension, Kingston, ON, Canada, 22–24 June 2011; pp. 91–100. [CrossRef]
67. Allan, V.H.; Kolesar, M.V. Teaching Computer Science: A Problem Solving Approach That Works. *SIGCUE Outlook* **1997**, *25*, 2–10. [CrossRef]

68. Dietz, G.; Landay, J.A.; Gweon, H. Building blocks of computational thinking: Young children's developing capacities for problem decomposition. In Proceedings of the Annual Meeting of the Cognitive Science Society, Montreal, QC, Canada, 24–27 July 2019; pp. 1647–1653 . Available online: https://api.semanticscholar.org/CorpusID:198232922 (accessed on 15 February 2024).
69. Westphal, B.; Harris, F.; Fadali, M. Graphical programming: A vehicle for teaching computer problem solving. In Proceedings of the 33rd Annual Frontiers in Education, Westminster, CO, USA, 5–8 November 2003; Volume 2. [CrossRef]
70. Brandt, J.; Dontcheva, M.; Weskamp, M.; Klemmer, S.R. Example-centric programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, GA, USA, 10–15 April 2010. [CrossRef]
71. Goldman, M.; Miller, R.C. Codetrail: Connecting source code and web resources. In Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, Herrsching, Germany, 15–19 September 2008; pp. 65–72. [CrossRef]
72. Kersten, M.; Murphy, G.C. Using Task Context to Improve Programmer Productivity. In Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Portland, OR, USA, 5–11 November 2006; pp. 1–11. [CrossRef]
73. Breese, J.S.; Heckerman, D.; Kadie, C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, USA, 24–26 July 1998; pp. 43–52; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1998.
74. Hill, W.; Stead, L.; Rosenstein, M.; Furnas, G. Recommending and Evaluating Choices in a Virtual Community of Use. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 7–11 May 1995; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1995; pp. 194–201. [CrossRef]
75. Shardanand, U.; Maes, P. Social Information Filtering: Algorithms for Automating "Word of Mouth". In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 7–11 May 1995; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1995; pp. 210–217. [CrossRef]
76. Balabanović, M.; Shoham, Y. Fab: Content-Based, Collaborative Recommendation. *Commun. ACM* **1997**, *40*, 66–72. [CrossRef]
77. GitHub and OpenAI. Github Copilot. 2023. Available online: https://github.com/features/copilot (accessed on 15 February 2024).
78. Dakhel, A.M.; Majdinasab, V.; Nikanjam, A.; Khomh, F.; Desmarais, M.C.; Ming, Z.; Jiang, J. GitHub Copilot AI pair programmer: Asset or Liability? *arXiv* **2023**, arXiv:2206.15331.
79. Xu, F.F.; Vasilescu, B.; Neubig, G. In-IDE Code Generation from Natural Language: Promise and Challenges. *ACM Trans. Softw. Eng. Methodol.* **2022**, *31*, 29. [CrossRef]
80. Jiang, E.; Toh, E.; Molina, A.; Olson, K.; Kayacik, C.; Donsbach, A.; Cai, C.J.; Terry, M. Discovering the Syntax and Strategies of Natural Language Programming with Generative Language Models. In Proceedings of the CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April–5 May 2022. [CrossRef]
81. Stumpf, S.; Sullivan, E.; Fitzhenry, E.; Oberst, I.; Wong, W.-K.; Burnett, M. Integrating Rich User Feedback into Intelligent User Interfaces. In Proceedings of the 13th International Conference on Intelligent User Interfaces, Gran Canaria, Spain, 13–16 January 2008; pp. 50–59. [CrossRef]