[[1]](#footnote-1)

Entwicklung einer immersiven Mixed-Reality-Anwendung zur virtuellen Tierbeobachtung und -fotografie

Silja-Marie Fischer, Julia Krauß

*Abstract*—In den letzten Jahren gewinnt Mixed Reality (MR) zunehmend an Bedeutung, besonders für Bildungs- und Unterhaltungsanwendungen. In dieser Arbeit wird ein Mixed-Reality-Prototyp zur virtuellen Tierbeobachtung auf der Meta Quest 3 entwickelt. Die Anwendung ermöglicht es, virtuelle Tiere in der realen Umgebung zu beobachten und zu fotografieren. Die Architektur folgt dem MVVM-3D-Muster, das Daten, Logik und Benutzeroberfläche klar voneinander trennt und gleichzeitig in der Library komplexere Berechnungen durchführt. Ein zentraler Service sorgt für die Verbindung zwischen Anwendung und Datenbasis. Der Prototyp zeigt, wie MR-Anwendungen mit realistischen Animationen, intuitiver Bedienung und raumbezogener Interaktion umgesetzt werden können.

*Index Terms*— Mixed Reality, Virtual Wildlife Observation, Virtual Interaction, Interactive Learning.

# Einleitung

D

ie wachsende Digitalisierung und häufige Nutzung mobiler Geräte haben das Lernen und die Wahrnehmung von Informationen stark beeinflusst. Untersuchungen zeigen, dass die Aufmerksamkeitsspanne in der Gesellschaft zunehmend kürzer wird [1]. Interaktive Lernmethoden, insbesondere Gamification und Extended Reality (XR), können allerdings die Motivation der Nutzer steigern und fördern langfristiges Lernen. Gamifizierte Elemente wie Herausforderungen, Belohnungen und interaktive Elemente sorgen für eine spielerische und gleichzeitig effektive Wissensvermittlung [2]. Mixed-Reality-Anwendungen (MR) können zudem das psychische Wohlbefinden fördern, indem sie virtuelle Naturerfahrungen schaffen. Ähnlich wie echte Naturaufenthalte können so Stress reduziert und kognitive Leistungen gesteigert werden [3].

Im Bereich Tierbeobachtung und Naturerfahrung existieren bisher kaum Anwendungen, die edukativen Mehrwert, spielerisches Erleben und Interaktion sinnvoll kombinieren. Viele VR-Lösungen bieten nur eingeschränkte Bewegungsmöglichkeiten und Interaktion. Mixed-Reality-Ansätze, die virtuelle Tiere in reale Umgebungen einbinden, gibt es bisher nur selten und vereinfacht.

Dieses Projekt zielt also darauf ab, eine prototypische MR-Anwendung zu entwickeln, die Nutzern ermöglicht, Tiere zu beobachten, mit ihnen zu interagieren und Fotos zu sammeln. Ein digitales Entdeckertagebuch soll Informationen zu den Tieren bereitstellen. Die Anwendung soll Unterhaltung mit Bildung kombinieren, das Interesse an biologischer Vielfalt wecken und das Bewusstsein für Naturschutz stärken.

# Stand der Technik

## Extended Reality

Extended Reality (XR), oder auch erweiterte Realität, beschreibt die Erweiterung der realen Welt um digitale Elemente [4]. Diese umfasst Augmented Reality (AR), Mixed Reality (MR) und Virtual Reality (VR) [5]. Ziel ist es, eine immersive Umgebung zu schaffen, in der das Bewusstsein von der physischen Welt entkoppelt ist [6]. VR ersetzt die Realität vollständig durch computergenerierte Welten, AR ergänzt die reale Umgebung um virtuelle Objekte, und MR erlaubt zudem Interaktionen zwischen realen und virtuellen Elementen [5,7].

XR wird in Werbung, Bildung und Unterhaltung eingesetzt, insbesondere aufgrund von Interaktivität, Multisensorik und erhöhter Emotionalität [5], [6], [7].

## Head-Mounted Displays

Head-Mounted Displays (HMDs) sind visuelle Ausgabegeräte, die am Kopf getragen werden und virtuelle Projektionen direkt vor den Augen erzeugen, wodurch eine 360°-Ansicht ermöglicht wird [X]. Sie kommen in Gaming, Ausbildung, Industrie und Marketing zum Einsatz [X]. HMDs lassen sich in drei Typen unterteilen: See-Through-HMDs erweitern die reale Umgebung mit digitalen Inhalten, Look-Around-HMDs beanspruchen nur einen Teil des Sichtfeldes, sodass die reale Welt weiterhin sichtbar bleibt, und Non-See-Through-HMDs isolieren den Nutzer vollständig [X]. HMDs nutzen stereoskopische Darstellungen, bei denen für jedes Auge ein leicht versetztes Bild angezeigt wird, wodurch ein Eindruck von Tiefe und Raum entsteht [X]. Technisch bestehen sie aus Displays, Linsen, Kameras, Sensoren, Prozessor, Speicher, Batterie sowie Controllern [X]. Das Tracking erfolgt über Inside-Out- oder Outside-In-Systeme [X], Hand- und Fingerbewegungen werden durch Controller, Handschuhe oder optische Systeme erfasst [X], Eye- und Gesichtstracking übertragen Blick und Mimik [X]. Zusätzliche Technologien wie Passthrough [X], Audiotechnik und haptisches Feedback erhöhen die Immersion und die Interaktivität [X].

TABelle I

Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| Kategorie | Anforderung | | Priorität |
| *Umgebung* | | Nutzer bewegt sich frei auf begehbarer Strecke | *Muss* |
| *Umgebung* | | Verschiedene Umgebungen (z. B. Wald, Savanne) | *Kann* |
| *Umgebung* | | Jede Umgebung enthält einzigartige Tiere | *Muss* |
| *Umgebung* | | Unterschiedliche Tageszeiten und Lichtstimmungen | *Kann* |
| *Tiere* | | Realistisches Verhalten (z. B. flüssige Bewegungen, Flucht) | *Sollte* |
| *Tiere* | | Tiere nur zu bestimmten Tageszeiten aktiv | *Sollte* |
| *Tiere* | | Tiere erscheinen zufällig | *Sollte* |
| *Tiere* | | Auftreten hängt von Tierart ab | *Kann* |
| *Kamera* | | Tiere mit virtueller Kamera fotografieren | *Muss* |
| *Kamera* | | Kamera mit Zoom, Fokus, Belichtung | *Kann* |
| *Kamera* | | Blitz für Nachtaufnahmen | *Kann* |
| *Tagebuch* | | Fotos werden gespeichert | *Muss* |
| *Tagebuch* | | Steckbrief mit Infos zu jedem Foto | *Muss* |
| *Tagebuch* | | Kombination aus echten und animierten Bildern | *Sollte* |
| *Tagebuch* | | „Seltene Entdeckungen“ markieren | *Kann* |
| *Tagebuch* | | Personalisierung mit Nutzername | *Kann* |
| *Plattform* | | App für Meta Quest 3 | *Muss* |
| *Benutzeroberfläche* | | Intuitive Steuerung von Kamera und Tagebuch | *Sollte* |
| *Benutzeroberfläche* | | Wischgesten im Tagebuch | *Kann* |
| *Account* | | Account erstellen | *Muss* |
| *Account* | | Login | *Muss* |
| *Account* | | Eigenes Entdeckertagebuch | *Muss* |
| *Mixed Reality* | | Virtuelle Tiere in echter Umgebung | *Muss* |
| *Mixed Reality* | | Virtuelle Umweltelemente in echter Umgebung | *Sollte* |
| *Mixed Reality* | | Interaktion mit Umgebung | *Kann* |
| *Belohnungssystem* | | Fotos werden bewertet | *Kann* |
| *Belohnungssystem* | | Seltene Tiere bringen mehr Punkte | *Kann* |
| *Belohnungssystem* | | Neue Kamerafunktionen/Umgebungen freischalten | *Kann* |
| *Missionssystem* | | Fotografie-Herausforderungen | *Kann* |
| *Missionssystem* | | Fortschritt im Tagebuch dokumentieren | *Kann* |
| *Erweiterungen* | | Regelmäßige Updates | *Kann* |
| *Multiplayer* | | Kooperativer Multiplayer-Modus | *Kann* |

*Muss*: Unverzichtbare Funktionen für den Prototypen

*Soll*: Wichtige, aber nicht zwingend erforderliche Funktionen

*Kann*: Optionale Funktionen zur Erweiterung des Erlebnisses

## Virtuelle Tierbeobachtung

In der virtuellen Tierbeobachtung kommen XR-Technologien in Form von AR-Einblendungen (WWF) [15], VR-Erlebnissen (Immotion) [16] oder vollständig virtuellen Zoos zum Einsatz [17], [18]. Vorteile sind Gefahrenfreiheit, Ortsunabhängigkeit und die Möglichkeit, ausgestorbene Arten realitätsnah darzustellen [16], [17], [19], [20]. Aktuelle XR-Spiele für Meta zeigen verschiedene Ansätze. National Geographic Explore VR erlaubt das Beobachten und Fotografieren von Tieren in natürlichen Umgebungen, ist dabei jedoch nur begrenzt interaktiv [7,8]. BRINK Traveler fokussiert sich auf realistisches Erkunden von Landschaften [9], Nature Treks VR auf Entspannung [10]. Ocean Rift bietet Unterwasserwelten und -tiere mit Lerninhalten [11,12], WildXR zeigt 360°-Videos realer Tiere, jedoch ohne aktive Interaktion [13]. AR- und MR-Spiele wie Living Room oder Animalz VR ermöglichen die Platzierung von Tieren in der eigenen Umgebung, meist ohne edukativen Mehrwert [14,15]. ZOSU Zoo kombiniert hingegen virtuelle Zoo-Umgebungen und Mixed-Reality und wird kontinuierlich erweitert [16].

Die Analyse zeigt, dass VR-Anwendungen im Bereich Natur- und Tiererfahrung am weitesten entwickelt sind, während AR- und MR-Anwendungen oft an realistischen Darstellungen, Interaktivität oder edukativem Nutzen fehlen. Daraus ergibt sich ein Potenzial für die geplante Mixed-Reality-Anwendung, die virtuelle Tiere realitätsnah in die eigene Umgebung integriert, aktive Beobachtung und Fotografie ermöglicht und die Erlebnisse in einem digitalen Entdeckertagebuch dokumentiert.

# Anforderungen

Tabelle 1 gibt einen Überblick über die Anforderungen der MR-Anwendung und deren Priorisierung in Muss, Soll und Kann. Sie bildet die Grundlage für Planung und Entwicklung.

# Technologieauswahl

Die Wahl der passenden Technologien ist entscheidend für die erfolgreiche Umsetzung der Anwendung. Da die Hardware bereits durch die Verfügbarkeit an der Hochschule festgelegt war, lag der Fokus auf der Auswahl der geeigneten Entwicklungsumgebung, Programmiersprache und des Architekturmodells.

## Hardware

Für die Entwicklung wurde die Meta Quest 3 genutzt, die über einen Snapdragon XR2 Gen 2-Prozessor, hochauflösendes Passthrough durch Farbkameras und Tiefensensoren sowie ein 4K-Display mit bis zu 120 Hz verfügt. Zusätzlich ermöglichen Pancake-Linsen, integrierte Lautsprecher und Handtracking eine hohe Immersion. Damit ist die Brille sowohl für VR- als auch MR-Anwendungen geeignet und bildet die technische Grundlage des Projekts [X].

## Entwicklungsumgebung

Um einen Prototypen entwickeln zu können, ist eine Entwicklungsumgebung nötig, welche die Erstellung von MR-Anwendungen ermöglicht. Im Folgenden werden Unity und Unreal Engine betrachtet [X].

Unity bietet eine Entwicklung mit C#, wird als einsteigerfreundlich beschrieben und ermöglicht schnelles Prototyping. Die Einarbeitung wird durch eine große Community und gute Dokumentation unterstützt. Dank Asset Store, Plugins und guter Integration für die Meta Quest 3 eignet sich Unity besonders für MR-Anwendungen [X]. Nachteilig ist eine geringere Grafikqualität im Vergleich zu Unreal.

Die Unreal Engine ermöglicht eine Entwicklung mit C++ und bietet eine hohe Leistung für grafisch hochwertige Spiele. Allerdings ist sie komplexer, schwieriger im Einstieg und mit höheren Kosten verbunden [X].

Da Unity geeigneter für die Entwicklung von Prototypen ist und guten Support für die genutzte Brille bietet, wurde diese Entwicklungsumgebung gewählt.

## Programmiersprache

Die Wahl der Programmiersprache hängt von Umgebung und Zielplattform ab. In Unity wird vor allem **C#** genutzt, das durch einfache Syntax, automatische Speicherverwaltung und eine große Community die Entwicklung von Prototypen erleichtert [X]. In der Unreal Engine dominiert **C++**, das durch manuelle Speicherverwaltung hohe Leistung ermöglicht, aber mehr Erfahrung erfordert. Mit Blueprints sind auch Anwendungen ohne Code per Drag-and-Drop umsetzbar [X]. Für dieses Projekt ist die Kombination aus Unity und C# optimal, während C++ eher für komplexe, leistungsorientierte Anwendungen geeignet ist.

## Architekturmodell

Zur Reduktion von Komplexität und zur klaren Trennung von Zuständigkeiten wird eine Aufteilung in Frontend und Backend umgesetzt. Im Frontend laufen alle Funktionen auf der Brille, wie Szenen, animierte Tiere und Interaktionen, während Backend Datenverarbeitung und -bereitstellung übernimmt. Dies verbessert Lesbarkeit, Wartbarkeit und Testbarkeit des Codes [X].

Für das Backend wird das MVC-Modell verwendet, da wenige Schnittstellen bestehen und die Entkopplung von UI und Logik ausreichend ist. Das Frontend ist dagegen UI- und interaktionslastig. Hier wird MVVM genutzt, da Data Binding reaktive Änderungen der View bei Datenänderungen ermöglicht und so Interaktionen zwischen realen und virtuellen Elementen einfach synchronisiert werden können [X]. Das klassische MVVM wird um eine eigene Library-Schicht erweitert, die komplexe XR-Funktionen wie Tracking, Physikberechnungen und Gestenerkennung kapselt. Zusätzlich enthält die View Komponenten, die auf unterschiedliche Eingaben wie Controller oder Handgesten reagieren. Die Trennung von Frontend und Backend unterstützt kollaborative Anwendungen und erlaubt eine strukturierte, wartbare Implementierung. Dieses erweiterte MVVM-Modell bildet die Basisarchitektur für das Frontend [X].

# Konzept

## Aufbau der Anwendung

Die Benutzeroberfläche startet mit der Login-Szene, in der sich Nutzer anmelden oder registrieren können. Nach erfolgreichem Login erscheint das Hauptmenü („Start“) mit den Optionen Strecke wählen, Konto, Tagebuch und Abmelden. Über die Kontoszene können persönliche Daten bearbeitet, gespeichert oder das Konto gelöscht werden; eine Rückleitung zur Login-Szene ist möglich.

In der Strecke-Szene wählt der Nutzer verschiedene Umgebungen wie Wald, Regenwald oder Antarktis aus. Jede Strecke enthält eigene Tiere und Umgebungselemente, die zufällig. Die Tiere verhalten sich realistisch. Ein Reh flieht bei Annäherung, eine Schnecke bewegt sich unbeirrt. Im Mixed-Reality-Kontext wird die Interaktion mit der realen Umgebung berücksichtigt, sodass virtuelle Tiere korrekt von realen Elementen verdeckt werden und auf echtem Untergrund stehen.

Die Kamerafunktion blendet eine virtuelle Kamera ins Sichtfeld ein, mit der Fotos aufgenommen und automatisch im Tagebuch gespeichert werden. Jeder Eintrag zeigt Name, Lebensraum, realistische Abbildung, Aufnahmefoto und Datum. Neue Aufnahmen erweitern das Tagebuch kontinuierlich. Das Spiel kann jederzeit über das Hauptmenü beendet werden. Die Umsetzung der beschriebenen Funktionen erfordert eine durchdachte Softwarearchitektur, um animierte Tiere realitätsnah in den physischen Raum zu integrieren und Interaktionen mit der Umgebung zu ermöglichen.

## Grundlegende Architektur

Die Architektur ist in drei Schichten gegliedert: Datenbasis, Backend und Frontend. Die Datenbasis verwaltet die Informationen unabhängig vom restlichen Programm, was parallelen Zugriff und flexible Speicherwahl ermöglicht. Das Backend übernimmt Zugriffskontrolle, Datenverarbeitung und Spiellogik und stellt die Daten über standardisierte APIs (Application Programming Interface) mit dem Hypertext Transfer Protocol (HTTP) bereit. Das Frontend ist für die Benutzeroberfläche zuständig und enthält ausschließlich Präsentationslogik.

Die Trennung reduziert Komplexität, verbessert Wartbarkeit und erleichtert Erweiterungen sowie eine mögliche Skalierung oder Multiplayer-Erweiterung.

### *Datenbank*

### In einer produktiven Anwendung übernimmt die Datenbank eine zentrale Rolle als „Single Source of Truth“ und muss Konsistenz, Integrität, Verfügbarkeit sowie Sicherheit gewährleisten. Für den Prototyp liegt der Fokus jedoch auf der Architektur und den interaktiven Aspekten der MR-Anwendung, nicht auf dauerhafter Datenpersistenz oder Performanz. Daher wird die Datenhaltung durch lokal gespeicherte CSV-Dateien simuliert, die leicht lesbar, versionierbar und plattformunabhängig nutzbar sind. Diese Struktur erlaubt eine spätere Migration in relationale Datenbanken. Bilder und Videos werden ebenfalls lokal gespeichert, könnten in produktiven Umgebungen jedoch in Cloud- oder Objekt-Datenbanken ausgelagert werden.

### *Backend*

### Das Backend folgt dem MVC-Muster und wird durch ein Repository ergänzt, das den Datenbankzugriff kapselt. Dadurch entsteht eine modulare und skalierbare Architektur. Das Repository übernimmt das Laden, Speichern und Löschen von Datensätzen und wandelt diese in nutzbare Objekte um. Das Model definiert die Datenstrukturen wie etwa eine Klasse „User“ mit ID, Name und E-Mail. Die View liegt in Form von REST-API-Endpunkten vor, über die Funktionen und Daten bereitgestellt werden. Der Controller koordiniert den Datenfluss zwischen den Schichten, enthält zusätzliche Logik wie das Anlegen von Accounts und übernimmt bei Erweiterungen zentrale Aufgaben wie Validierung und Synchronisierung. Durch die klare Trennung bleiben Anpassungen auf einzelne Schichten beschränkt, was Wartbarkeit und Erweiterbarkeit sicherstellt.

### *Frontend*

### Das Frontend basiert auf dem Architekturmodell MVVM-3D nach Stonis [X], das die klassischen MVVM-Komponenten Model, View, Library und ViewModel für XR erweitert. Das ViewModel übernimmt die zentrale Steuerung, empfängt Änderungen von View oder Model und synchronisiert diese über Data Binding. Properties verknüpfen UI-Elemente mit Variablen und gewährleisten eine automatische Aktualisierung zwischen Anzeige und Datenbasis.

### Die klare Trennung erlaubt zudem die Einbindung von Logiken aus der Library. Nutzeraktionen werden von der View an das ViewModel übergeben, dort verarbeitet und an das Model sowie Backend weitergeleitet, Änderungen fließen zurück in die View. Für Mehrbenutzerszenarien wurde MVVM-3D zu MVVM-3DC [X] erweitert. Grundlage ist der Centralized Mode, bei dem das Backend als Server und mehrere Frontend-Instanzen als Clients agieren. Über SignalR kann eine bidirektionale Kommunikation zwischen Server und Clients hergestellt werden, sodass Nutzeraktionen in Echtzeit synchronisiert werden. In dieser Arbeit wird MVVM-3D prototypisch umgesetzt, MVVM-3DC jedoch nur konzeptionell betrachtet.

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

Abb. 1. MVVM-3D (eigene Darstellung)

## Interaktion

Die Interaktion erfolgt über die Meta Quest 3 mit Controllern, Handtracking und Texteingaben. Controller dienen zur Navigation, Aktionen und Fotografie, Handtracking ermöglicht natürliche Gesten, und Texteingaben unterstützen Anmeldung und Personalisierung. Die Umgebung basiert auf einem vordefinierten, kollisionsfähigen Mesh, also einem 3D-Gitter aus Polygonen [X]. Dieses simuliert Boden, Wände und Hindernisse, sodass Tiere realistisch auf Annäherung, Kollisionen oder Hindernisse reagieren können. Ein statischer Raum gewährleistet stabile Performance, da kontinuierliches Scanning zu hohem Rechenaufwand führen würde. Das Konzept kombiniert multimodale Eingaben, eine begehbare Umgebung, tierisches Verhalten und Fotografie im digitalen Entdeckertagebuch. Es ist zugleich für Erweiterungen wie Raumscanning oder Multiplayer vorbereitet.

# Prototypenentwicklung

In den folgenden Unterkapiteln werden zentrale Aspekte der Prototypimplementierung erläutert, insbesondere MVVM-3D und Interaktionen.

## Datenbank

Für die Datenbasis wurden fünf CSV-Dateien lokal gespeichert und über GitHub versioniert, sodass ein Austausch möglich war, ohne ein produktives System zu nutzen. Alle Daten liegen in Textform vor und können später in eine relationale Datenbank überführt werden. Abbildung 2 zeigt das ER-Diagramm mit den Klassen *Animal*, *User*, *Diary*, *Diaryentry* sowie der Verbindungsklasse *DiaryDiaryentry*. Die Klasse *Animal* enthält Informationen zu den virtuellen Tieren (ID, Name, Lebensraum, Pfade zu 3D-Objekten und Fotos). Medien werden nur über Pfade referenziert. *User* speichert ID, Name, Username, Mail und Passwort. Für ein produktives System ist hier ein externer Dienst wie Firebase sinnvoll. *Diary* umfasst ID und User und kann um Personalisierungseinstellungen erweitert werden. *Diaryentry* verknüpft Tagebücher mit Tieren. Da ein Tagebuch mehrere Einträge und ein Eintrag mehrere Tagebücher enthalten kann, wurde die Verbindungsklasse *DiaryDiaryentry* eingeführt. Diese enthält zusätzlich Datum und Pfad zu aufgenommenen Fotos, wodurch personalisierte Einträge ohne Redundanz möglich sind. Die Struktur trennt generische Inhalte von nutzerspezifischen Informationen, bleibt erweiterbar und erleichtert die spätere Migration in eine relationale Datenbank.

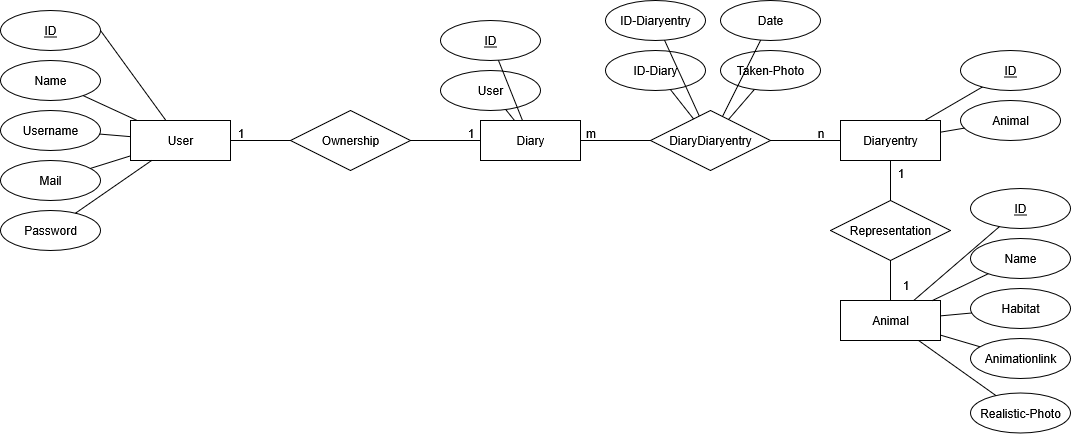


Abb. 2. ER-Diagramm Datenbank (eigene Darstellung)

## Backend

Das Backend wurde in C# mit LINQ-Ausdrücken und dem Framework ASP.NET Core implementiert. Alle Repositories folgen demselben Aufbau. Sie binden CSV-Dateien ein, lesen deren Inhalte zeilenweise ein und mappen die Spaltenwerte auf Attribute von C#-Objekten. Änderungen erfolgen durch Rückschreiben der Attributwerte in die entsprechenden Zellen. Welche Objekte geladen werden, wird über LINQ-Ausdrücke gesteuert, die SQL-ähnliche Abfragen im Code ermöglichen.

Die Models bilden ausschließlich die Datenstruktur ab und stellen Getter- und Setter-Methoden bereit. Über Controller werden die Repositories angebunden und die Daten dem Frontend bereitgestellt. Mit ASP.NET Core können dafür REST-Endpunkte definiert werden, etwa „GetAll“ zum Laden aller Objekte oder „Create“ für die Erstellung eines neuen Tieres. Beim Anlegen eines Tieres wird automatisch eine neue ID vergeben und parallel ein verknüpfter Tagebucheintrag erzeugt. Löschanfragen löschen zunächst alle zugehörigen Einträge, bevor das Tier entfernt wird, um Konsistenz und Speicheroptimierung zu gewährleisten.

## Frontend

Für die Einrichtung der Unity-Umgebung auf der Meta Quest 3 wird Unity über den Unity Hub installiert. Da das Headset auf Android basiert, müssen Anwendungen dafür als APK-Dateien bereitgestellt werden. Dazu wird im Unity Editor das Build-Target auf Android gesetzt und der Entwicklermodus aktiviert. Die Übertragung erfolgt per USB-C über „Build and Run“, wobei Unity die APK automatisch auf dem Gerät installiert und startet. So lassen sich Anwendungen schnell testen und anpassen.

Die Implementierung des Frontends erfolgt in C# mit Unity-spezifischen Befehlen. Inhalte werden in „Scenes“ organisiert, UI-Elemente im Inspector angepasst und mit Variablen oder Events verknüpft. Models spiegeln die Backend-Daten wider und werden per SerializeField serialisierbar gemacht. Services kapseln die Kommunikation mit dem Backend über einen zentralen HTTP-Client und geben Objekte an die ViewModels weiter. Die ViewModels implementieren die Logik zwischen View und Model, verwalten Daten in ObservableCollections und sorgen über PropertyChanged für die automatische UI-Aktualisierung. Komplexere Berechnungen werden in einer Library ausgelagert, die zudem eine Brücke zu Unity-spezifischen Funktionen (MonoBehaviour) bereitstellt. Die View-Schicht bindet UI-Elemente an ViewModel-Funktionen und ist damit vollständig Unity-spezifisch. Szenenwechsel oder einfache Aktionen werden direkt in der View umgesetzt, während komplexe Logiken über das ViewModel angestoßen werden. Um Daten wie den aktuellen User über Szenenwechsel hinweg verfügbar zu machen, wird eine SessionData-Klasse genutzt, die sich auch für Erweiterungen wie Spielstände eignet.

## Interaktion

Die Interaktion im Prototypen wurde mit dem Meta All-in-One Software Development Kit umgesetzt. Dieses stellt unter anderem Building Blocks für Controllersteuerung, Handtracking, Passthrough und kollisionsfähige Meshes bereit [X]. Für die Umgebung wird das Mixed Reality Utility Kit in Kombination mit Effect Mesh genutzt. Dadurch entsteht ein statisches, kollisionsfähiges Mesh, das Wände, Boden und Hindernisse abbildet. Virtuelle Tiere reagieren realistisch auf die Umgebung, und die Performance bleibt stabil. Passthrough sorgt dafür, dass reale Elemente weiterhin sichtbar sind. Controller erlauben präzise Eingaben. Handtracking erkennt Gesten wie Zeigen, Greifen oder Wischen. Auf der Brille lassen sich Buttons und Eingabefelder jedoch aktuell nicht anwählen, da der sogenannte Ray nicht sichtbar ist. Im Unity-Editor ist dies mit Maus und Tastatur allerdings möglich. Zum Testen wurde ein Würfel eingesetzt, der sich sowohl mit Controllern als auch per Handtracking greifen lässt.

Zukünftig soll eine virtuelle Tastatur die Texteingabe direkt auf der Brille ermöglichen, wenn die Eingabefelder und Buttons anwählbar sind. Dies ist besonders für Login und das digitale Entdeckertagebuch relevant. Langfristig könnten Interaktionen auch in einem Multiplayer-Kontext synchronisiert werden. So könnten mehrere Nutzer gleichzeitig dieselbe Szene erleben und gemeinsam mit den virtuellen Tieren interagieren.

# Zusammenfassung und Ausblick

## Die vorliegende Arbeit demonstriert die Umsetzung eines Mixed-Reality-Prototyps für die virtuelle Tierbeobachtung auf der Meta Quest 3. Durch die Verwendung der MVVM-3D-Architektur konnte eine klare Trennung zwischen Daten, Logik und Darstellung erreicht werden. Controller, Handtracking und kollisionsfähige Meshes ermöglichen eine interaktive, immersive Nutzung. Data Binding sorgt dafür, dass Änderungen in Echtzeit in der Ansicht sichtbar werden, was besonders für Fotografie und Echtzeitinteraktionen relevant ist. Trotz bestehender Einschränkungen bei der Bedienung von UI-Elementen auf der Brille zeigt der Prototyp das Potenzial von Mixed-Reality-Anwendungen für Lern- und Spielerlebnisse. Die Nutzung von Building Blocks des Meta SDK beschleunigt die Implementierung und bietet eine stabile Grundlage für zukünftige Erweiterungen.

## Zukünftige Entwicklungen könnten qualitativ hochwertigere Tieranimationen, eine virtuelle Tastatur zur direkten Texteingabe, dynamisches Raumscanning oder Multiplayer-Funktionen umfassen. Auch die Verbesserung der Reaktionsfähigkeit durch optimiertes Data Binding eröffnet neue Möglichkeiten für Interaktion und Fotografie.

## Insgesamt liefert der Prototyp einen soliden Ausgangspunkt für die Weiterentwicklung immersiver, interaktiver Tierbeobachtungsanwendungen.

Literaturverzeichnis

[1] Max-Planck-Institut für Bildungsforschung, „Mit der Informationsflut sinkt die Aufmerksamkeitsspanne der Gesellschaft“. Zugegriffen: 16. August 2025. [Online]. Verfügbar unter: https://www.mpib-berlin.mpg.de/pressemeldungen/informationsflut-senkt-aufmerksamkeitsspanne

[2] E. Ratinho und C. Martins, „The role of gamified learning strategies in student’s motivation in high school and higher education: A systematic review“, *Heliyon*, Bd. 9, Nr. 8, S. e19033, Aug. 2023, doi: 10.1016/j.heliyon.2023.e19033.

[3] M. G. Berman *u. a.*, „Interacting with nature improves cognition and affect for individuals with depression“, *J. Affect. Disord.*, Bd. 140, Nr. 3, S. 300–305, Nov. 2012, doi: 10.1016/j.jad.2012.03.012.

1. Diese Arbeit wurde am 30. August 2025 im Rahmen des Forschungsprojekts zu Mixed-Reality-Anwendungen für Tierbeobachtung und -fotografie unter der Betreuung von Prof. Dr.-Ing. Sascha Seifert und M.Sc. Max Barchet eingereicht. S.-M. Fischer und J. Krauß sind Studierende des M.Sc. Studiengangs Advanced Information Technology (MIT) an der Fakultät für Technik der Hochschule Pforzheim, Tiefenbronner Str. 66, 75175 Pforzheim, Deutschland.

   (E-Mail: fische2l@hs-pforzheim.de, kraussju@hs-pforzheim.de) [↑](#footnote-ref-1)