R3-A4. Erstellung eines adaptiven und immersiven Virtual-Reality (VR)-Lernpfads.

Erasmus+

EINFÜHRUNG

Die Entwicklung hin zu einer integrativeren und anpassungsfähigeren Ausbildung im Natursteinsektor nimmt mit der Schaffung eines hochgradig immersiven Virtual-Reality (VR)-Lernpfads Gestalt an. Dieses Projekt ist ein wesentlicher Bestandteil der Integration fortschrittlicher Technologien in die Ausbildung und erleichtert so eine personalisierte und effektive Lernumgebung.

Die Gestaltung dieses Bildungsweges erfordert einen detaillierten Entwicklungsprozess, der von der Identifizierung der Schlüsselszenarien bis zu ihrer Umsetzung in einer VR-Umgebung reicht. Er beginnt mit einer Forschungs- und Analysephase, in der die repräsentativsten Situationen und Aufgaben des Sektors sorgfältig ausgewählt werden. Anschließend wird ein didaktisches Design entwickelt, das sowohl die Funktionalität als auch die Zugänglichkeit berücksichtigt und sicherstellt, dass jede Phase der Ausbildung für alle Nutzer relevant und erreichbar ist.

Mit diesem Ansatz sollen die VR-Szenarien nicht nur die Arbeitsbedingungen genau simulieren, sondern auch auf die spezifischen Bildungsbedürfnisse der Nutzer abgestimmt werden. Die daraus resultierende immersive Erfahrung zielt nicht nur darauf ab, zu lehren, sondern auch die Nutzer zu befähigen, praktische Fähigkeiten in einer sicheren und kontrollierten Umgebung zu entwickeln, um sie in der realen Welt anwenden zu können.

Letztendlich wird dieses Bildungsinstrument nicht nur zu einer Brücke zur beruflichen Kompetenz, sondern auch zu einem Mittel zur Förderung eines besseren Verständnisses und einer größeren Akzeptanz von Vielfalt am Arbeitsplatz.

Dieser Bericht und alle Informationen über das Projekt sind auf der Website von InclusiveStone verfügbar: https://inclusivestone.eu/

Inhalt

1. WERKZEUGENTWICKLUNG

Der Kern dieses Projekts basiert auf der immersiven Fähigkeit, die die virtuelle Realität bietet. Derzeit basiert der Standard für die Erstellung von Anwendungen in diesem Bereich auf der Verwendung von Grafik-gesteuerten Anwendungen, die aus der Welt der Videospiele stammen. Aus diesem Grund wurde für die Entwicklung unseres Bildungswerkzeugs eine Methodik gewählt, die der Entwicklung eines Videospiels ähnelt.

Diese Methodik beruht auf drei grundlegenden Säulen, die in den folgenden Abschnitten näher erläutert werden:

- **Szenengestaltung**
- **Entwicklung der 3D-Umgebung.**
- **Entwicklung von Funktionen und immersiven Erfahrungen.**

Um die Entwicklung effizient und effektiv durchzuführen, wurden verschiedene Technologien eingesetzt:

- **Blender:** Ein kostenloses, plattformübergreifendes Software-Tool, das für eine breite Palette von 3D-Grafikfunktionen wie Modellierung, Animation und Rendering verwendet wird. Dank seines Open-Source-Charakters und der großen Community ist es durch Tutorials und eine umfangreiche Dokumentation leicht zu erlernen.
- **Unity:** Dieses ebenfalls quelloffene Grafikdesign-Tool ist in der Videospielentwicklung weit verbreitet. Es zeichnet sich durch seine Skalierbarkeit und die Möglichkeit der Anpassung durch Skripte aus, unterstützt durch eine robuste Community und eine vollständige Dokumentation.
- **Oculus Quest:** Ein leicht zugängliches und einfach zu bedienendes Virtual-Reality-Gerät, das zu einer der wichtigsten Stützen der Zukunftsvision von Meta geworden ist. Es bietet den Vorteil, dass es kabellos ist und sich nahtlos in Unity integrieren lässt, wodurch die Zugänglichkeit und Vielseitigkeit bei der Entwicklung von VR-Anwendungen verbessert wird.

2. SZENENGESTALTUNG

Bei diesem Verfahren geht es um die detaillierte Planung der Abfolge von Schritten, die der Nutzer vom Beginn seiner Interaktion mit dem Tool bis zu dem Moment, in dem er ein vollständiges immersives Erlebnis erreicht, durchführen wird. Kurz gesagt, es geht um die Erstellung des Skripts, das die umfassende Entwicklung der App leiten wird.

Zunächst wurden 6 Arbeitsszenarien ausgewählt, die die verschiedenen im Arbeitspaket R1 identifizierten Rollen angemessen repräsentieren, da sie anpassungsfähig sind und keine Gefahr für Menschen mit unterschiedlichen Behinderungen darstellen. Die ausgewählten Situationen sind wie folgt:

- **Gabelstaplerfahrer - Gütertransport:** Bei dieser Aufgabe geht es darum, drei Paletten mit Marmorplatten auf- und abzuladen und sie von der Verarbeitungsanlage zum vorgesehenen Lagerbereich zu transportieren. Diese Tätigkeit umfasst das Manövrieren in Bereichen mit begrenztem Raum und das Stapeln von Materialien mit hohem Gewicht.
- **Gabelstaplerfahrer - LKW-Beladung:** In diesem speziellen Szenario besteht das Ziel darin, ein Transportfahrzeug mit drei Paletten Marmorplatten zu beladen, ausgehend von einem Lagerbereich. Der Vorgang erfordert die Durchführung von Manövern auf engem Raum und das Verstauen von schweren Ladungen.
- **Bediener eines Brückenkrans - Handhabung von Platten:** Bei dieser Aufgabe hat der Benutzer die Aufgabe, 2 Marmorplatten in den beleuchteten Lagerbereich zu transportieren, wobei er die erforderlichen Manöver ausführt und die Sicherheitsvorschriften beachtet.
- **Bediener eines Brückenkrans - Handhabung von Blöcken:** Bei dieser Tätigkeit besteht die Aufgabe des Benutzers darin, einen Marmorblock vom Lkw zur Ladefläche der Verarbeitungsmaschine zu transportieren, wobei er die erforderlichen Manöver ausführt und die geltenden Sicherheitsvorschriften strikt beachtet.
- **Bediener der Reinigungsanlage:** Zweck dieser Tätigkeit ist es, die geeignete Persönliche Schutzausrüstung (PSA) für die Reinigung und Desinfektion der Arbeitsanlage zu ermitteln und auszuwählen, um Verschmutzungen und Ablagerungen zu beseitigen. Dies muss verantwortungsbewusst und mit größtmöglichem Respekt für die Umwelt erfolgen.
- **Abfallwirtschaft:** Bei der Entwicklung dieser Aufgabe ist der Benutzer für das Management von Chemikalien und Abfällen verantwortlich. Dies umfasst die geeignete Auswahl von Techniken und Werkzeugen für das Recycling und die korrekte Entsorgung von Abfällen sowie die Einhaltung von Leitlinien, die eine verantwortungsvolle und ökologisch nachhaltige Abfallwirtschaft gewährleisten sollen.

In diesem Zusammenhang wurde beschlossen, ein Szenariomodell zu entwerfen, das aus aufeinanderfolgenden Etappen oder Missionen im Stil von Videospielen besteht, d. h. man kann erst dann zur nächsten Etappe übergehen, wenn man die vorherige abgeschlossen hat, und zwar schrittweise. Auf diese Weise wird dem Benutzer die Notwendigkeit auferlegt, den gesamten Prozess von Anfang bis Ende sicher abzuschließen, und jede Unfähigkeit, dies zu tun, wird als Misserfolg im Szenario gewertet.

Innerhalb dieser Reihe von Phasen besteht die erste Phase aus einem detaillierten Tutorial über die spezifische Maschine oder Aufgabe. Auf diese Weise wird es für den Benutzer einfacher, die Bedienelemente für jede Situation zu erlernen, was eine flüssigere Bedienung ermöglicht, ohne dass er sie häufig überprüfen muss. Um mit den spezifischen Aufgaben fortzufahren, die von der jeweiligen Situation abhängen.

2.1. Tutorial

Im Laufe der Designphase der VR-App haben wir ein grundlegendes Element zur Orientierung und Vorbereitung des Spielers implementiert: das Tutorial. Diese Komponente ist essentiell, da sie eine strukturierte und methodische Einführung in die Spielumgebung, die verfügbaren Werkzeuge und die Missionsmechanik bietet.

Das Tutorial-Panel wurde sorgfältig entwickelt, um eine umfassende Lernerfahrung zu gewährleisten. Es begann mit der Erstellung eines detaillierten Skripts, das jeden kritischen Schritt aufschlüsselt, den der Spieler verstehen muss, bevor er mit der Mission beginnt.

Der erste Abschnitt des Tutorials befasst sich mit der Handhabung und den Funktionen der Maschinen, die die Spieler während des Spiels bedienen werden. Anschließend stellt das Tutorial das Teleportationswerkzeug vor, eine entscheidende Mechanik, die die strategischen und mobilen Möglichkeiten im Spiel erweitert. Es wurden spezielle Übungen entwickelt, die es den Spielern ermöglichen, mit diesem Werkzeug in einer kontrollierten Umgebung zu experimentieren, um sicherzustellen, dass sie es beherrschen, bevor sie sich Echtzeitsituationen stellen.

Darüber hinaus enthält das Dashboard Informationsmodule über die Nutzung zusätzlicher Funktionen, die für den Auftrag relevant sind. Jedes Element wurde sorgfältig erläutert und mit praktischen Beispielen versehen, um ein ganzheitliches Verständnis für seine Nutzung und Anwendung zu gewährleisten.

Abbildung 1: Tutorial zum Panel

Die Realisierung dieses Tutorials war einer der Eckpfeiler bei der Entwicklung unseres Videospiels, um zu gewährleisten, dass jeder Benutzer über die notwendigen Kenntnisse und Erfahrungen verfügt, um das Spiel in vollem Umfang zu genießen und seine Missionen und damit die eigentliche Arbeit erfolgreich zu bewältigen.

2.2. Missionen

Bei der Entwicklung unseres Projekts haben wir ein Missionssystem implementiert, das für den Fortschritt des Benutzers beim Erlernen der Arbeit grundlegend ist. Dieses System ist so konzipiert, dass es für verschiedene Rollen und Aufgaben geeignet ist.

Die Missionen lassen sich in zwei Hauptkategorien einteilen, die zur Vielfalt des Tools beitragen:

- **Action-Missionen**: Diese Missionen sind das Herzstück der Interaktivität im Spiel. Sie erfordern eine aktive Beteiligung des Spielers und die Erfüllung von Aufgaben, die von der Bedienung von Maschinen bis hin zum Einsatz der Hände beim Aufheben und Kontrollieren von Reinigungsarbeiten reichen können. Diese Missionen sind so konzipiert, dass sie die motorischen und strategischen Fähigkeiten des Spielers testen.
- **Q&A-Missionen**: In diesen Missionen wird der Spieler mit Situationen konfrontiert, die ein Nachdenken und eine Entscheidungsfindung auf der Grundlage der in den Kursen erhaltenen Informationen erfordern, die der Benutzer haben muss, damit

diese App wirklich nützlich ist. In diesen Missionen werden das Verständnis und das erworbene Wissen des Benutzers durch Frage- und Antwortfelder bewertet.

Abbildung 2:Block Transportauftrag

Jede Mission, unabhängig von der Kategorie, ist sorgfältig mit der nächsten verknüpft, um sicherzustellen, dass die Erfahrung jedes Spielers zusammenhängend und tiefgreifend ist. Außerdem fördert dieses Missionsdesign eine natürliche Lernkurve, bei der die Spieler ihre Fähigkeiten im Laufe des Spiels verbessern können.

Kurz gesagt, unser Anspruch an den Nutzen soll nicht nur unterhalten, sondern den Spieler auch in einen Prozess des kontinuierlichen Lernens und Entdeckens einbinden, was für das Gesamterlebnis, das wir bieten wollen, wesentlich ist.

3. ENTWICKLUNG DER 3D-UMGEBUNG

Die Erstellung von Werkzeugen in Grafik-gesteuerten Anwendungen für Videospiele beinhaltet häufig die Handhabung von Objekten. Daher besteht die zweite Hauptaufgabe in der Herstellung der verschiedenen Objekte oder Anlagen, die sowohl direkt als auch indirekt in den zuvor erstellten technischen Skripten festgelegt werden. Dieser Vorgang gliedert sich in drei wesentliche Phasen:

- **Modellierung**: Eine mathematische Darstellung eines Objekts wird mit Hilfe einer speziellen Software in drei Dimensionen erstellt. Für dieses Projekt haben wir uns für die Verwendung von Blender entschieden.
- **Aufbau**: In dieser Phase werden die Bestandteile eines Objekts in ein kontrollierbares digitales Skelett zerlegt, was die Erstellung flüssiger und präziser Animationen erleichtert. Dieser Schritt ist für Objekte, die animiert werden müssen, von entscheidender Bedeutung und wurde ebenfalls mit Blender durchgeführt.
- **Texturierung**: Bei der Texturierung werden Farben und Details auf die 3D-Modelle aufgetragen, um ihnen ein realistischeres und detaillierteres Aussehen zu verleihen, ebenfalls mit der Software Blender.

Die für dieses Projekt hergestellten Objekte decken ein breites und vielfältiges Spektrum ab, das von extrem einfachen Elementen bis zu komplexen und detaillierten Maschinen reicht. Die produzierten Objekte wurden in verschiedene Kategorien eingeteilt, je nach ihrer Komplexität und Funktion im Spiel. Wir können finden:

- Behälter, Gebäude, Regale und dekorative Elemente.

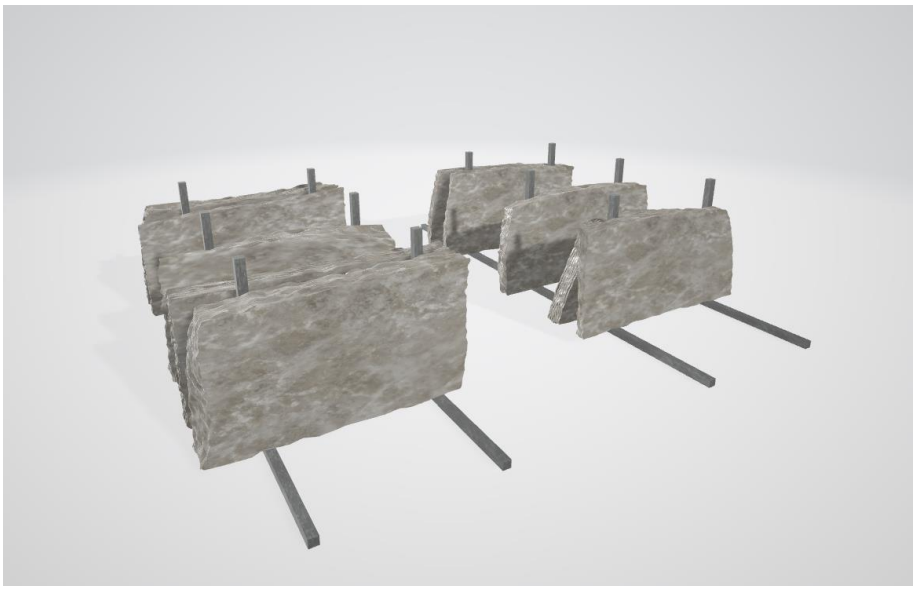


Abbildung 3: Steinplatten.

- Persönliche Ausrüstung und nützliche Gegenstände, die eine gewisse Funktionalität aufweisen.

Abbildung 4: Andere Elemente.

- Spezielle Maschinen für verschiedene Situationen.

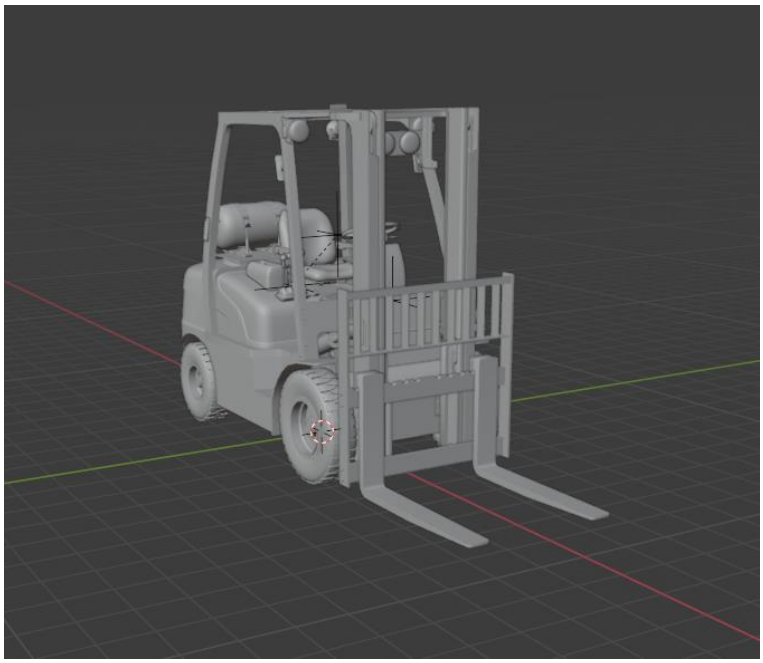


Abbildung 5: Gabelstapler-Modell

Sobald die Objekte, die im Spiel verwendet werden sollen, entworfen und erstellt sind, ist der nächste Schritt die Vorbereitung der Szenarien. Dieser Prozess ist von entscheidender Bedeutung für die Entwicklung der Funktionalität des Spiels, um die Umgebung, in der die endgültige Handlung stattfindet, genau wiederzugeben. Wichtig ist, dass es sich um einen schrittweise wiederholenden Prozess handelt, der immer wieder angepasst werden muss, da

sich bei späteren Tests die Notwendigkeit von Änderungen herausstellen kann. Für die Erstellung des Szenarios wurde die Grafik-gesteuerte Anwendung verwendet, die auch bei der Produktion der endgültigen Anwendung zum Einsatz kommen wird.

Da es sich bei den in den vorangegangenen Aufgaben ausgewählten Stellen um Positionen handelt, die in der Fabrik und nicht im Steinbruch ausgeübt werden, wurde nur ein Szenario entwickelt. Das für dieses Projekt entwickelte Szenario sieht wie folgt aus:

- **Natursteinfabrik**: Die Bühne ist ein virtuelles Abbild eines realen Werks mit klar abgegrenzten Bereichen wie dem Maschinenraum, dem Chemielager und dem Bereich der Abfallentsorgung. Diese detaillierte Gestaltung hilft den Nutzern, sich mit einer spezifischen Arbeitsumgebung und den verschiedenen Aufgaben in der Natursteinindustrie vertraut zu machen.

Abbildung 6: Werksstufe

4. ENTWICKLUNG VON FUNKTIONEN UND IMMERSIVEN ERFAHRUNGEN

Die Entwicklung der Funktionalität ist der Schritt, in dem die zuvor gesammelten Konzepte und Informationen zum Leben erweckt werden. In dieser Phase geht es darum, das theoretische Werkzeug in eine praktische und einsatzfähige Anwendung zu verwandeln. Für diese Aufgabe

haben wir uns für die Grafik-gesteuerte Anwendung entschieden, da sich mit ihr Virtual-Reality-Elemente leicht einbinden lassen und die Nutzergemeinde umfangreiche Unterstützung bietet.

In Unity sind Projekte in Szenen unterteilt, die durch eine Reihe von Ereignissen miteinander verbunden werden können. Jede Szene besteht aus mehreren Objekten, denen spezifische Komponenten zugewiesen werden, die ihre Attribute und ihr Verhalten im Spiel definieren. Der Entwicklungsprozess für die Funktionalität ist sehr umfangreich und kann in mehrere Phasen unterteilt werden, darunter **App-Design und -Architektur**, **Datenmodellierung** zur Strukturierung von Informationen, **SDK-Integration** zur Erweiterung der Anwendungsfunktionen und **Ereignisverwaltung** für einen interaktiven und konsistenten Spielfluss.

4.1. Design und Architektur

Design und Architektur beinhalten die detaillierte Ausarbeitung des zuvor entwickelten Drehbuchs, um den Weg, den der Benutzer gehen muss, und die Entscheidungen, die in jeder Szene oder auf jedem Bildschirm des Spiels zu treffen sind, genau zu definieren. Dieser Prozess ist von entscheidender Bedeutung, da er die Grundstruktur festlegt, auf der das Benutzererlebnis aufgebaut wird. In dieser Phase werden die Navigationslogik, die Anordnung der interaktiven Elemente und die Abfolge der Ereignisse, die den Benutzer durch das Spiel führen sollen, berücksichtigt. Die Interaktion mit den einzelnen Objekten und Charakteren in der virtuellen Umgebung sowie die Übergänge zwischen den Szenen werden sorgfältig geplant, um ein intuitives und immersives Erlebnis zu gewährleisten, das den Benutzer mit der Geschichte und den Zielen des Spiels vertraut macht.

Innerhalb des Projekts werden die Benutzer auf zwei Arten von in Unity entworfenen Hauptszenen stoßen: die Menüszene und die spezifische Szene für jede Arbeitsstation.

4.1.1. Name des Auftraggebers

Der Bildschirm "Menü" ist eine intuitive und zugängliche Einführung in die virtuelle Umgebung, in der die Benutzer ihre ersten Schritte mit dem Tool unternehmen. Zu Beginn müssen sie eine grundlegende Entscheidung treffen: die Spracheinstellungen, um sicherzustellen, dass die Erfahrung personalisiert und für sie verständlich ist. Sobald diese grundlegende Präferenz festgelegt ist, führt der Bildschirm den Benutzer intuitiv zu der Szene, nämlich der Wahl einer bestimmten Arbeitssituation, die er erkunden möchte. Obwohl die Interaktivität in dieser Phase bewusst eingeschränkt ist, um den Fokus auf das Lernziel zu richten, ist die Erfahrung so gestaltet, dass sie einladend und orientierend wirkt und den Benutzer auf den Übergang zur nächsten Phase vorbereitet, in der die im Menü getroffene Auswahl die spezifische Arbeitsszene bestimmt, die als nächstes aufgerufen wird.

4.1.2. *Arbeitsstatus Szene*

Diese Phase stellt den Kern des Tools dar, in der jede vorgeschlagene Arbeitssituation in ihrer eigenen individuellen Szene zum Leben erweckt wird. Nach dem zuvor festgelegten Missionsschema tauchen die Benutzer in eine Reihe von Aufgaben ein, die die Verantwortlichkeiten und Herausforderungen einer bestimmten Stelle nachbilden sollen. Das Erlebnis beginnt mit einem detaillierten Tutorial, das genau auf die jeweilige Aufgabe abgestimmt ist und eine solide Grundlage für das Erlernen und Ausführen der nachfolgenden Missionen bietet.

Wenn der Benutzer fortschreitet und jede Mission besteht, wird die nächste freigeschaltet, was einen linearen Fortschritt und ein Erfolgsgefühl vermittelt. Nach erfolgreicher Bewältigung der letzten Mission haben Sie die Möglichkeit, die Szene neu zu starten und Ihre Fähigkeiten zu verbessern oder zum Hauptmenü zurückzukehren, um einen andere Tätigkeiten zu erkunden und kennenzulernen, was einen kontinuierlichen Lernprozess und eine vielfältige Erkundung der unterschiedlichen Facetten der Natursteinbranche ermöglicht.

4.2. Datenmodellierung

Das Ziel dieser Phase ist es, einen übergreifenden Rahmen für die Speicherung und den Abruf von Daten zu schaffen. Wir haben uns für PlayerPrefs entschieden, ein Unity-Modul, das die Speicherung von Informationen durch eine Datenverschlüsselung erleichtert, die mit dem Namen des Projekts verbunden sind. In Oculus-Anwendungen werden diese Variablen in Form einer XML-Datei gespeichert, die sich im Verzeichnis /data/data/pkg-name/shared_prefs/pkg-name.v2.playerprefs.xml befindet, wobei "pkg-name" dem Namen entspricht, der der

Anwendung zugewiesen wurde. Unity macht den Zugriff auf diese Daten durch die Funktionen der Klasse PlayerPrefs extrem einfach.

Aufgrund der Struktur dieser Anwendung gibt es eine gemeinsame Variable, die wir speichern müssen: diejenige, die die Sprache bestimmt. Wir werden die folgenden PlayerPrefs-Funktionen verwenden, um sie zu verwalten:

- **SetString (*Name, Wert*):** Mit dieser Funktion wird der mit "Name" bezeichneten Variablen ein Textwert zugewiesen, der dem ISO-Code der gewählten Sprache entspricht.
- **GetString (*Name*):** Mit dieser Funktion können wir den der Variablen "Name" zugewiesenen Wert abrufen. Wir werden sie in jeder Szene verwenden, um die ausgewählte Sprache zu identifizieren.

4.3. SDK-Integration

Die Erstellung von Virtual-Reality-Anwendungen basiert auf drei Grundpfeilern: einem kompatiblen Gerät, das an einen Computer angeschlossen werden kann, einer Videospiel-Grafik-Engine und einem Software Development Kit (SDK). Zu Beginn dieses Kapitels wurden die ersten beiden Elemente ausführlich beschrieben, wobei Oculus das Gerät und Unity die gewählte Grafik-Engine ist. Für das SDK, d. h. die Tools, die den Softwareentwicklern bei der Erstellung helfen, gibt es mehrere Optionen, die für Unity geeignet sind. Obwohl es ein spezifisches SDK für Oculus gibt, wurde das XR Interaction Toolkit ausgewählt, ein von Unity entwickeltes Paket, das die Anpassung des Projekts an verschiedene Gerätetypen erleichtert. Das bedeutet, dass das Tool zwar ursprünglich für Oculus Quest entwickelt wurde, aber auf jedem mit Unity kompatiblen Gerät installiert werden kann, wodurch seine Zugänglichkeit und Reichweite erweitert wird.

Um das SDK zu integrieren und die Interaktion in Unity zu aktivieren, führen Sie die folgenden Schritte aus:

1. Wenn Sie Unity starten, erstellen Sie ein neues Projekt mit der Universal Render Pipeline-Vorlage, die der Schlüssel zu optimierten Grafiken ist - ein entscheidender Aspekt für ein reibungsloses und zufriedenstellendes VR-Erlebnis.
2. Das SDK, in diesem Fall das XR Interaction Toolkit, wird über das Fenster Package Manager installiert, das unter Windows integriert ist.
3. Es ist notwendig, den Gerätetyp anzugeben, auf dem das Projekt entwickelt werden soll. Aktivieren Sie dazu das Kontrollkästchen VR unterstützt in Projekteinstellungen/Player.

Mit diesen Schritten ist Ihr Unity-Projekt bereit für die Entwicklung von Virtual Reality. Um jedoch mit der Interaktion Ihres Geräts mit der grafischen Umgebung zu beginnen, müssen Sie unbedingt die Anweisungen im nächsten Abschnitt befolgen.

4.4. Ereignisverwaltung

Die Ereignisverwaltung ist wohl der umfangreichste Teil bei der Erstellung der Funktionalität unserer App. Es basiert auf der Verwendung der Komponenten, die Unity bietet, zusammen mit der Erstellung von prägnanten Skripten, die die gewünschte Interaktion zwischen dem Benutzer und dem System ermöglichen. Da jede Szene einzigartige Details enthält, die sich auf die Interaktion mit ihr auswirken, könnte dieser Abschnitt noch erheblich erweitert werden. In diesem Dokument werden wir uns jedoch darauf beschränken, die allgemeinen und entscheidenden Aspekte hervorzuheben, die für das Erreichen unserer Interaktivitätsziele unerlässlich sind.

4.4.1. Die Kulisse für VR-Interaktion schaffen

Sobald Sie Ihr Unity-Projekt und die grafisch gestalteten Szenarien fertiggestellt haben, besteht der nächste Schritt darin, die Szene für das Interaktionsgerät vorzubereiten, um sie mit einer Kamera zu integrieren und so das Testen während der Entwicklung zu erleichtern. Dieses Verfahren, das in den folgenden Schritten beschrieben wird, muss für jede der Szenen im Projekt ausgeführt werden:

1. Entfernt die Standardkamera, die standardmäßig in der Szene erscheint.
2. Erstellen Sie ein leeres Objekt und fügen Sie ihm eine XR-Rig-Komponente hinzu, die als Kern der Interaktion zwischen dem Benutzer und der Maschine fungieren wird. Wir werden dieses Objekt VR-Rig nennen.
3. Innerhalb des VR-Rig wird ein leeres untergeordnetes Objekt erstellt, das als anfänglicher Referenzpunkt für die Benutzerinteraktion dient. Wir nennen es Kamera-Offset.
4. Fügen Sie eine Kamera als untergeordnetes Objekt des Kamera Offset-Objekts hinzu und fügen Sie eine Tracked Pose Driver-Komponente zu dieser Kamera hinzu. Dieses Element wird als unser virtuelles Auge fungieren und die Komponente wird die Drehung der Ansicht ermöglichen.
5. Konfigurieren Sie die Variablen der XR-Rig-Komponente (übergeordnetes Rig, VR-Rig) mit der Referenzposition und der Kamera, die Sie gerade erstellt haben. Stellen Sie den Wert "floor" im Attribut "Tracking Origin Mode" so ein, dass sich die Kamera automatisch an die Höhe des Benutzers anpasst.
6. Um die Mobilität und Sichtbarkeit des Fahrers zu ermöglichen, erstellen Sie zwei zusätzliche Objekte als untergeordnete Objekte des Kamera-Offsets und statten sie mit dem Attribut XR-Controller aus.
7. Weisen Sie das Modell, das Sie als Controller verwenden möchten, im Attribut Model Prefab der einzelnen XR-Controller zu.
8. Schließlich wird diesen beiden Objekten die Komponente XR Direct Interactor hinzugefügt, die die Möglichkeit bietet, mit anderen Objekten in der Szene zu interagieren.

4.4.2. Controller-Eingang

Dieser Prozess bezieht sich auf die Erfassung der Werte, die bei der Interaktion mit den Schaltflächen auf den Steuerelementen ausgegeben werden. Diese Werte sind entscheidend für die Durchführung einer Vielzahl von Aktionen innerhalb der App, wie das Greifen von Objekten oder die Auswahl verschiedener Optionen. Bevor mit diesen Daten gearbeitet wird, ist es wichtig zu verstehen, welche Art von Informationen die einzelnen Eingaben auf den Steuerelementen liefern. Diese Informationen können im XR Interaction Debugger-Fenster eingesehen werden, das über Window/Analysis zugänglich ist.

Sobald Sie dies umgesetzt haben, besteht der nächste Schritt darin, ein Skript zu entwickeln, mit dem Sie diese Werte erfassen können. Die zu diesem Zweck erstellte Gruppe heißt `HandController` und wird als Komponente an jedes der 3D-Modelle angehängt, die im Attribut Model Prefab der Steuerelemente angegeben sind. Dise sind nachfolgende:

```
public class HandController : MonoBehaviour
{
    //Input
    private InputDevice targetDevice;

    //Controller model options
    public bool showController = false;
    public List<GameObject> controllers;
    private GameObject spawnedController;

    //Device characteristics
    public InputDeviceCharacteristics controllerChar;

    //Hand model
    public GameObject handModel;
    private GameObject spawnedHandModel;

    private Animator handAnimator;

    //Get if is Right or left
    public string isRight;

    //Input variables
    public float trigVal;
    public bool primaryButton;
    public float gripValue;
    public Vector2 axisVal;
```

Abbildung 7: Gruppe Hand Controller

1. Am Anfang des Skripts können Sie die Deklaration (Name/Typ) der Variablen sehen. Diejenigen, die öffentlich sind, können geändert werden und fungieren als Attribute der Komponente.
2. Wenn Sie ein neues Skript erstellen, generiert Unity automatisch zwei Funktionen: `Start`, die nur einmal zu Beginn ausgeführt wird, wenn die Komponente instanziiert wird, und `Update`, die während des Lebenszyklus der Komponente wiederholt ausgeführt wird.
3. Die erste Funktion, die innerhalb dieser Komponente ausgelöst wird, ist `TryInitialize`, die das VR-Gerät anhand seiner spezifischen Merkmale identifiziert, z. B. ob der Controller rechts oder links ist, indem sie die Funktion `GetDevicesWithCharacteristics` des `InputDevices-Moduls` verwendet.

```
//Try get input device and attach the selected model to it
void TryInitialize()
{
    //Get VR devices by characteristics
    List<InputDevice> devices = new List<InputDevice>();
    InputDevices.GetDevicesWithCharacteristics(controllerChar, devices);


    if (devices.Count > 0)
    {
        //Get target device and get a controller model
        targetDevice = devices[0];
        GameObject prefab = controllers.Find(con => con.name == targetDevice.name);

        if (prefab)
        {
            spawnedController = Instantiate(prefab, transform);
        }
        else
        {
            Debug.LogError("Did not find corresponding controller model");
            spawnedController = Instantiate(controllers[0], transform);
        }

        //Instantiate Hand Model and get Animator component
        spawnedHandModel = Instantiate(handModel, transform);
        handAnimator = spawnedHandModel.GetComponent<Animator>();
    }
}
```

Abbildung 8: TryInitialize-Funktion

4. Die Funktion `Update` wird dann ausgeführt und ruft die `Funktionen UpdateEvents` und `UpdateAnimations` auf.

5. `UpdateEvents` sammelt die Werte, die von den Steuerelementen mit der Funktion `TryGetFeatureValue` ausgegeben werden, und speichert sie in den vorher deklarierten öffentlichen Variablen.

6. `UpdateAnimations` ist eine Funktion, die dazu dient, die Animationen für die einzelnen Tasten zu aktivieren, wenn ein Handmodell vorhanden ist. Obwohl diese Funktion für die Erfassung von Werten nicht entscheidend ist, trägt sie zur Interaktivität und zum Realismus der Aktionen bei.

```
//Update hand events
void  UpdateEvents()
{
    //Trigger event
    if(targetDevice.TryGetFeatureValue(CommonUsages.trigger, out float triggerVal))
    {
        trigVal = triggerVal;
    }

    //PrimaryButton event
    if (targetDevice.TryGetFeatureValue(CommonUsages.primaryButton, out bool primary))
    {
        primaryButton = primary;
    }

    //Grip event
    if (targetDevice.TryGetFeatureValue(CommonUsages.grip, out float gripVal))
    {
        gripValue = gripVal;
    }

    //Joystick event
    if(targetDevice.TryGetFeatureValue(CommonUsages.primary2DAxis, out Vector2 axis))
    {
        axisVal = axis;
    }
}
```

Abbildung 9: Funktion UpdateEvents

4.4.3. Staplersteuerung

Um eine überzeugende und funktionelle Virtual-Reality-Erfahrung bei der Bedienung eines Gabelstaplers zu erreichen, ist eine Reihe von detaillierten und spezialisierten Skripten erforderlich. Diese Skripte sind für die präzise Interaktivität und Steuerung verantwortlich, die erforderlich sind, um den Betrieb dieser Art von Maschinen effektiv zu simulieren. Innerhalb des VR-Tools wird ein primäres Skript für das Fahren benötigt, das es dem Benutzer ermöglicht, den Gabelstapler innerhalb der virtuellen Umgebung zu manövrieren. Darüber hinaus werden mehrere zusätzliche Skripte benötigt, die sich auf die Steuerung der Hebel und anderer

Mechanismen des Staplers konzentrieren, um sicherzustellen, dass alle möglichen Aktionen im realen Betrieb virtuell nachgebildet werden können.

Im Folgenden werden wir jedes dieser Skripte im Detail erläutern und untersuchen, wie sie funktionieren, wie sie mit den Elementen des Gabelstaplers interagieren und wie sie zu einer immersiven und realistischen Benutzererfahrung beim Umgang mit dieser Maschine im Kontext der virtuellen Realität beitragen.

- Das CarController-Skript bietet ein interaktives Fahrerlebnis in der virtuellen Realität, bei dem der Benutzer die Beschleunigung des Fahrzeugs durch das Auslösen der Steuerelemente direkt beeinflussen kann, was sich in der Variable `accelVal` widerspiegelt. Darüber hinaus passt das Skript die Richtung des Fahrzeugs an, indem es den Winkel der Räder auf der Grundlage von Benutzereingaben verändert, was über die Variable `currentSteerAngle` abgewickelt wird. Die Bremsen werden über eine Schaltfläche aktiviert, was sich auf die Variable `isBreaking` auswirkt, und das Skript verwaltet die Kollisionsphysik, um bei Bedarf ein Aufprallen zu simulieren. Kurz gesagt, dieser Code setzt Benutzerinteraktionen in mechanisches Fahrzeugverhalten innerhalb der virtuellen Umgebung um.

```
// Update is called once per frame
void FixedUpdate()
{
    GetValues();

    if (!isBouncing)
    {
        foreach (WheelCollider wheel in wheels)
        {
            wheel.motorTorque = strength * Time.deltaTime * accelVal;

            wheel.wheelDampingRate = dampening;

            wheel.brakeTorque = isBreaking ? brakeStrength : 0;
        }

        foreach (var wheel in wheelsSteer)
        {
            WheelCollider collider = wheel.GetComponent<WheelCollider>();
            Transform trans = wheel.transform;

            collider.steerAngle = currentSteerAngle;
            collider.wheelDampingRate = dampening;
            //UpdateWheelVisual(collider, trans);
        }
    }

}
```

Abbildung 10: FixedUpdate-Funktion

- Mit dem Skript `RotateElement kann` der Benutzer die Drehung des Aufzugs in der Virtual-Reality-Umgebung durch manuelle Steuerung kontrollieren. Durch Interaktion mit der vertikalen Eingabeachse des Controllers passt die `Variable upForce die` Intensität und Richtung der Drehung des toRotate-Objekts an und bestimmt sie. Mit eingebauten Beschränkungen, die die Drehung stoppen, wenn vordefinierte Grenzen erreicht werden, sorgt dieses Skript für einen reibungslosen und kontrollierten Betrieb des Drehmechanismus des Aufzugs.

```
void FixedUpdate()
{
    UpdateForce();

    if (upForce > 0 && rotateUp)
    {
        float rotationAmount = 20f * Time.deltaTime * upForce;
        toRotate.localRotation *= Quaternion.AngleAxis(rotationAmount, Vector3.up);
        rotateDown = true;
    }
    else if (upForce < 0 && rotateDown)
    {
        float rotationAmount = 20f * Time.deltaTime * upForce;
        toRotate.localRotation *= Quaternion.AngleAxis(rotationAmount, Vector3.up);
        rotateUp = true;
    }
    }

void OnTriggerEnter(Collider col)
{
    if(col.gameObject.tag == "RotationLimit" && upForce > 0)
    {
        rotateUp = false;
    }
    else if(col.gameObject.tag == "RotationLimit" && upForce < 0)
    {
        rotateDown = false;
    }
}

void UpdateForce()
{
    if (inputs.Length == 0)
    {
        inputs = GameObject.FindGameObjectsWithTag("GameController");
    }
    else
    {
        foreach (var element in inputs)
        {
            HandController hand = element.GetComponent<HandController>();

            if (hand.isRight == "right" && grab.isGrabbing)
            {
                upForce = hand.axisVal.y * 0.5f;
            }
            else if(!grab.isGrabbing)
            {
                upForce = 0;
            }

        }
    }
}
```

Abbildung 11: Clase RotateElement

- Die `Skripte` MoveUp `und` MoveRetractil arbeiten zusammen, um die vertikale Bewegung eines Objekts in einer Virtual-Reality-Umgebung zu steuern. Während `MoveUp die` direkte vertikale Bewegung des Objekts steuert, steuert `MoveRetractil` die vertikale

Bewegung einer zugehörigen Komponente, z. B. eines einziehbaren Mastes. Beide verwenden die vertikale Krafteingabe `upForce`, die durch die Interaktion des Benutzers mit dem Controller beeinflusst wird, um die Richtung und Intensität der Bewegung zu bestimmen. Die Koordination zwischen diesen beiden Skripten sorgt für eine reibungslose und eingeschränkte vertikale Bewegung, wobei `MoveRetractil` auch Kollisionen und räumliche Beschränkungen berücksichtigt, um unerwünschte Bewegungen zu verhindern, z. B. das Erreichen des oberen oder unteren Endes des zulässigen Weges.

```
// Update is called once per frame
void FixedUpdate()
{
    UpdateForce();

    if (goUp && upForce > 0)
    {
        transform.Translate(new Vector3(0, 0, 1f) * upForce * constForce);
        goDown = true;
    }
    else if (goDown && upForce < 0)
    {
        transform.Translate(new Vector3(0, 0, 1f) * upForce * constForce);
        goUp = true;
    }

}

void UpdateForce()
{
    if (inputs.Length == 0)
    {
        inputs = GameObject.FindGameObjectsWithTag("GameController");
    }
    else
    {
        foreach (var element in inputs)
        {
            HandController hand = element.GetComponent<HandController>();

            if (hand.isRight == "right" && grab.isGrabbing)
            {
                upForce = hand.axisVal.y * 0.5f;
            }
            else if (!grab.isGrabbing)
            {
                upForce = 0;
            }

        }
    }
}
```

Abbildung 12: Clase MoveUp

4.4.4. Brückenkran-Steuerung

Die Steuerung des Brückenkrans in der Virtual-Reality-Umgebung ist ein integraler Bestandteil des interaktiven Erlebnisses und ermöglicht es dem Benutzer, Objekte in einem dreidimensionalen Raum präzise und realistisch zu manipulieren. Für die Bewegung von Brücke und Haken werden zwei wesentliche Skripte verwendet: `CraneController` und `HookController`. Ersteres verwaltet die Bewegungen des Brückenkrans, wie z. B. die Verschiebung in verschiedene Richtungen, während letzteres sich mit den spezifischen Bewegungen des Hakens befasst und das Absenken, Aufsteigen und die präzise Positionierung des Hakens ermöglicht. Für das Greifen und die Handhabung von Brettern und anderen Objekten wird außerdem das Skript `AttachTarget` verwendet, das ein effektives An- und Abkoppeln ermöglicht und dafür sorgt, dass die Objekte während des Transports fixiert bleiben und am gewünschten Zielort reibungslos freigegeben werden. Zusammen bilden diese Skripte ein zusammenhängendes und effizientes Steuerungssystem für den Brückenkranbetrieb innerhalb der Simulation.

- Das Skript `CraneController` ist das Herzstück der Brückenkransteuerung und ermöglicht die seitliche, vertikale und frontale Bewegung von Brückenkranmotor, Kabine und Seilen. Es verwendet die Variable `power`, um die Intensität und Richtung der Bewegung zu bestimmen, während `state` die Art der ausgeführten Bewegung angibt. Boolesche Variablen (Wahrheitswerte) wie `isBack`, `isFront`, `isLeft`, `isRight`, `isDown` und `isUp` fungieren als Endschalter, um zu verhindern, dass sich der Kran über seine physikalischen Grenzen hinaus bewegt. Andererseits arbeitet der `HookController mit dem CraneController zusammen` und überwacht insbesondere Kollisionen mit bestimmten Tags, um die vertikale Bewegung des Hakens zu steuern und zu begrenzen und zu verhindern, dass er sich über die oberen und unteren Grenzen hinaus bewegt. Beide Skripte sorgen für einen reibungslosen und sicheren Betrieb des Brückenkrans, so dass der Benutzer Lasten in der virtuellen Umgebung präzise handhaben kann.
- Das `AttachTarget-Skript` kümmert sich um die Mechanik des Greifens und Haltens von Objekten, wie z. B. Brettern, in einer Virtual-Reality-Umgebung. Es verwendet ein `HingeJoint`, um einen flexiblen Verbindungspunkt zwischen dem Objekt und einem Festhaltepunkt zu simulieren, der eine kontrollierte Pendelbewegung ermöglicht. Die Erkennung der Nähe des Objekts zu einem Ziel und die Benutzereingabe sind entscheidend für die Aktivierung des Griffs. Wenn der Benutzer die Greiftaste drückt und sich das Objekt in der richtigen Position befindet, aktiviert das Skript die Fesseltextur und konfiguriert die Eigenschaften des `HingeJoints`, indem es Grenzen und Geschwindigkeit festlegt, um eine realistische Bewegung zu simulieren. Wenn der Benutzer die Greiftaste loslässt oder sich das Objekt zu weit entfernt, deaktiviert das Skript die Verbindung, sodass das Objekt kontrolliert losgelassen werden kann. Darüber hinaus interagiert das Skript mit einem Anweisungsfeld, um den Benutzer durch den Prozess der Handhabung der Objekte zu führen.

```
// Update is called once per frame
void FixedUpdate()
{
    //Check stop state and power value
    if (power != 0 && state != "Block")
    {
        //Lateral movement
        if (state == "left_right")
        {
            if (power > 0 && !isLeft)
            {
                motor.transform.Translate(new Vector3(1f, 0, 0) * power * constForce);
                isRight = false;
            }
            else if (power < 0 && !isRight)
            {
                motor.transform.Translate(new Vector3(1f, 0, 0) * power * constForce);
                isLeft = false;
            }
        }

        //Vertical movement
        if (state == "up_down")
        {
            if(power > 0 && !isUp)
            {
                ropes.transform.localScale = ropes.transform.localScale + new Vector3(0, 0, 1f) * (-power * constForce * scaleValue);

                foreach (var ele in gancho)
                {
                    ele.transform.Translate(new Vector3(0, 0, 1f) * power * constForce);
                }

                isDown = false;
            }
            else if (power < 0 && !isDown)
            {
                ropes.transform.localScale = ropes.transform.localScale + new Vector3(0, 0, 1f) * (-power * constForce * scaleValue);

                foreach (var ele in gancho)
                {
                    ele.transform.Translate(new Vector3(0, 0, 1f) * power * constForce);
                }

                isUp = false;
            }
        }

        //Frontal movement
        if (state == "back_front")
        {
            if (power > 0 && !isBack)
            {
                overhead.transform.Translate(new Vector3(0, 1f, 0) * power * constForce);
                isFront = false;
            }
            else if (power < 0 && !isFront)
            {
                overhead.transform.Translate(new Vector3(0, 1f, 0) * power * constForce);
                isBack = false;
            }
        }
    }
}
```

Abbildung 13: FixedUpdate-Funktion der Gruppe CraneController

4.4.5. Fragebogen-Manager

Um den Prozess der Erstellung von Fragebögen zu optimieren und zu vereinfachen, wurde eine Gruppe namens `QuizManager` entwickelt, die den Aufbau von Fragebögen standardisiert. Die Funktionsweise dieser Gruppe gliedert sich in die folgenden Schritte:

1. Es werden Variablen definiert, um alle Fragen, die aktuelle Frage, die richtige Antwort und Elemente der Benutzeroberfläche (UI) zu speichern.
2. Die `Funktion` InitQuiz `ist implementiert`, deren Zweck es ist, alle notwendigen Variablen zu initialisieren und dann `SetNextQuestion` aufzurufen, um das Quiz zu starten.

3. Die `Funktionen SetNextQuestion` und `SetAnswerOptions` sind dafür zuständig, die Variablen und Komponenten des Panels, wie Texte und Schaltflächen, zu aktualisieren, die Daten der neuen Frage zuzuweisen und gegebenenfalls die vorherige Frage zu verwerfen.
4. Die Funktion `SelectButton(Button but)` wird aktiviert, wenn eine Antworttaste gedrückt wird. Bei Fragen mit einer Antwort führt diese Funktion zur Ausführung von `CheckResponse`.
5. Die Funktion `CheckResponse` enthält die Logik, die erforderlich ist, um zu überprüfen, ob die ausgewählte Antwort richtig ist. Wenn ja, wird der Benutzer zum nächsten Feld weitergehen; andernfalls haben Sie die Möglichkeit, die Antwort erneut zu versuchen.

4.4.6. Übersetzer

Dieser Prozess wirkt auf jede der Szenen im Hintergrund, um jedes Element der Benutzeroberfläche des Tools mit in die ausgewählte Sprache übersetzten Texten zu versehen.

Der erste Schritt bestand darin, die Texte in die Sprachen der beteiligten Partner zu übersetzen. Diese Übersetzungen sind in einem Format strukturiert, in dem jedes Wort oder jeder Satz mit einem eindeutigen Schlüssel verknüpft ist, so dass es einfach ist, den Text in der gewünschten Sprache abzurufen. Um dieses System zu verwalten, wurden zwei spezielle Klassen entwickelt:

1. `MainTranslate`: Diese Klasse hat die Aufgabe, Übersetzungsskripte mit der Betriebsumgebung, d. h. dem Code, zu verknüpfen. Wann immer ein bestimmter Schlüssel, der in dem genannten Skript erzeugt wurde, lokalisiert werden muss, liefert `MainTranslate` das entsprechende Wort oder den entsprechenden Satz in der richtigen Sprache.
2. `SceneTranslate`: Diese Klasse hat die Aufgabe, `MainTranslate` zu Beginn jeder Szene mitzuteilen, welche Sprache im Optionsmenü ausgewählt wurde.

Wenn diese Prozesse aktiv sind, muss der Benutzer lediglich jede der einzufügenden Textzeilen durch die folgende Codezeile ersetzen:

```
text = MainTranslate.Fields[textKey];
```

Abbildung 14: Zeile übersetzen.

4.4.7. Interaktion mit Zeigestab

Diese Funktion ermöglicht es dem Benutzer, mit Objekten oder Tafeln zu interagieren, die sich in einer gewissen Entfernung befinden. Dazu wird ein virtueller Zeigestab erzeugt, der von der Hand des Benutzers ausgeht und es ihm ermöglicht, bestimmte Aktionen an bestimmten Objekten auszuführen. Im Rahmen dieses Projekts ist es wichtig, dass der Benutzer in der Lage

ist, die mit dieser Technik gestellten Fragen zu beantworten. Um diese Art der Interaktion zu realisieren, müssen die folgenden Schritte befolgt werden:

1. Erzeugen Sie für jede Hand ein Ray-Interactor-Objekt, das über das Menü GameObject/XR zugänglich ist.
2. Wählen Sie die Objekte, mit denen interagiert werden soll, indem Sie den Parameter Raycast Mask der Komponente XR Ray Interactive anpassen. Mit dieser Einstellung können Sie die Ebene auswählen, auf der der Stab wirken soll. Daher müssen die Objekte, die mit dem Stab interagieren sollen, der entsprechenden Ebene zugewiesen werden.
3. In diesem Szenario haben wir uns dafür entschieden, den UI-Layer für die von Unity bereitgestellten Panels auszuwählen.
4. Um zu verhindern, dass der Stab ständig sichtbar ist, muss das Attribut Ungültiger Farbverlauf der Komponente XR Interactor Line Visual auf einen transparenten Wert geändert werden. Auf diese Weise wird der Stab nur sichtbar, wenn die Hand auf ein Panel zeigt.
5. Beide Objekte werden als Unterobjekte des Kamera-Offset-Objekts eingefügt, das zuvor in der Vorbereitungsphase der Szene erstellt wurde, um sicherzustellen, dass sie Teil des mit dem VR-Gerät verbundenen Objekts sind.

Mit der Implementierung dieser Schritte wird die Funktionalität erreicht, die für die Erstellung und Interaktion mit den verschiedenen oben beschriebenen Dashboards erforderlich ist.

Abbildung 15: Interaktion mit Zeigestab

5. YOUTUBE-VIDEOS

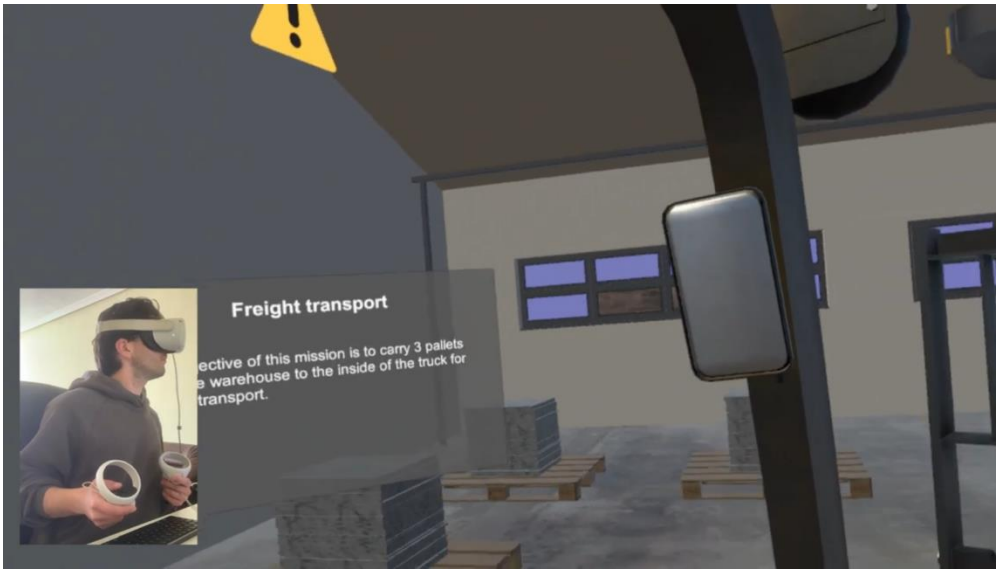
Um das White Paper zu ergänzen und eine dynamischere Perspektive auf unser VR-Tool zu bieten, haben wir eine Reihe von YouTube-Videos ausgewählt, die seine Funktionalität demonstrieren. Im Folgenden stellen wir diese audiovisuellen Materialien vor, die einen klaren und direkten Eindruck von der Leistung und den Möglichkeiten unserer VR-Lösung vermitteln.

- Förderung: https://www.youtube.com/watch?v=Ogs2WzzCRe0&ab_channel=AEIPiedraNatural

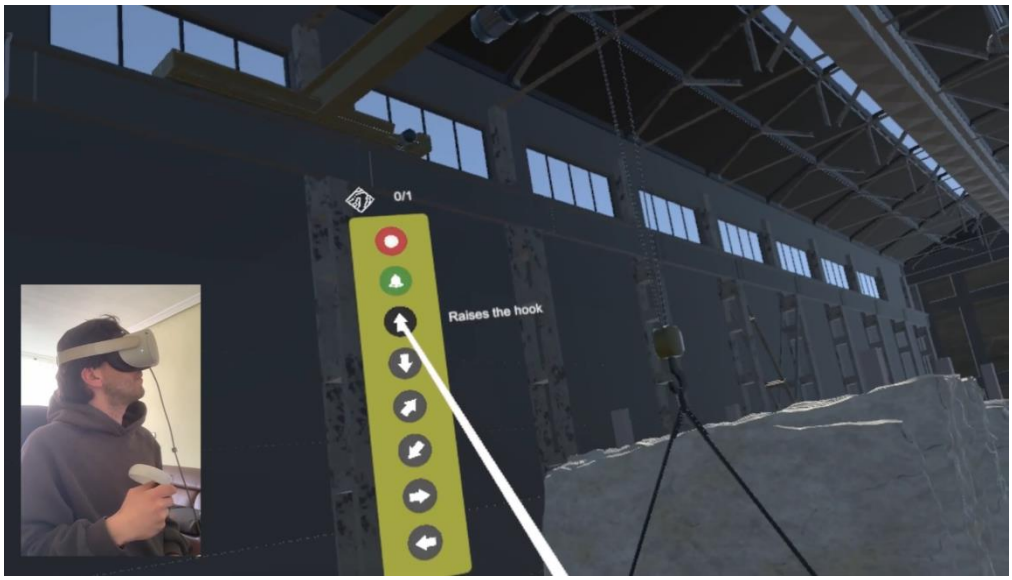
- Hauptmenü: https://www.youtube.com/watch?v=I7j9rTMeWmo&ab_channel=AEIPiedraNatural

- Gabelstapler - Lagerung: https://www.youtube.com/watch?v=hCkCl9ihiLU&t=14s&ab_channel=AEIPiedraNatural

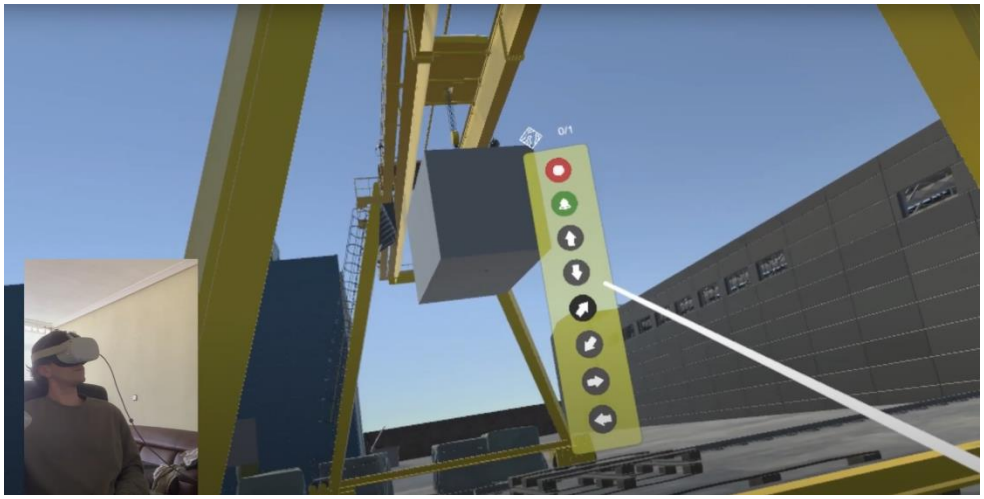
- Gabelstapler - Lkw-Beladung: https://www.youtube.com/watch?v=NnpA44V6DMY&t=13s&ab_channel=AEIPiedraNatural



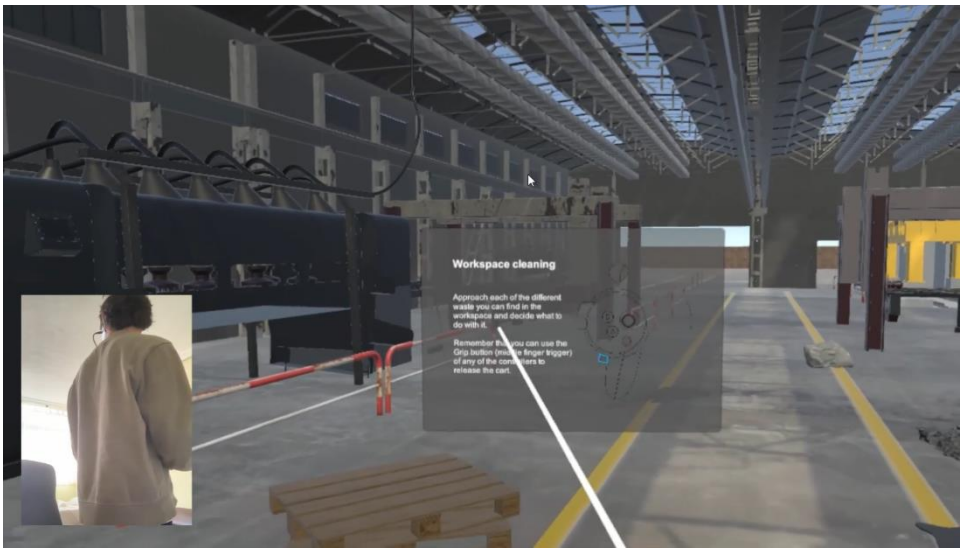
- Brückenkran - Brammen: https://www.youtube.com/watch?v=LK9pSCPLMrw&ab_channel=AEIPiedraNatural



- Brückenkran - Blöcke: https://www.youtube.com/watch?v=tVyFFRJyQ4U&ab_channel=AEIPiedraNatural



- Reinigung: hattops://www.youtube.com/watch?v=twiffarjyak4u&ab_channel=apedarentural



- Abfallwirtschaft: https://www.youtube.com/watch?v=mojMZ2G6Huc&ab_channel=AEIPiedraNatural