Uniwersytet Bielsko-Bialski

PRACA DYPLOMOWA

WYDZIAŁ
BUDOWY MASZYN I INFORMATYKI
KIERUNEK: **INFORMATYKA**
SPECJALNOŚĆ: **TECHNIKI TWORZENIA OPROGRAMOWANIA**

SZYMON BIAŁEK

nr albumu: 55872

Praca magisterska

PROJEKT ZASTOSOWANIA CZUJNIKÓW DO ODWZOROWANIA RUCHU KOŃCZYN DOLNYCH DLA ŚRODOWISKA WIRTUALNEJ RZECZYWISTOŚCI

Kategoria pracy: projektowa

Promotor: dr MARCIN BERNAŚ

Bielsko-Biała, r. ak. 2023/2024

Spis Treści

Wprowadzenie

Wirtualna rzeczywistość (VR) to zaawansowana technologia, która pozwala użytkownikom na interakcję z komputerowo-generowanym, trójwymiarowym środowiskiem w sposób imitujący rzeczywistość (1). Dzięki niej możliwe jest przeniesienie użytkownika do wirtualnych światów, które mogą odzwierciedlać zarówno rzeczywiste miejsca, jak i całkowicie fikcyjne scenariusze. VR znajduje szerokie zastosowanie w wielu dziedzinach, z których każda korzysta z jej unikalnych technologii (2).

W branży rozrywki i gier komputerowych, VR umożliwia tworzenie niezwykle immersyjnych i interaktywnych doświadczeń, które angażują użytkownika na niespotykaną dotąd skalę (3). Gracze mogą przenieść się do fantastycznych światów, wziąć udział w dynamicznych akcjach i przygodach, a wszystko to z poczuciem realności i obecności w wirtualnym środowisku.

W edukacji VR oferuje nowe możliwości nauki przez doświadczenie. Uczniowie mogą wirtualnie odwiedzać historyczne miejsca, eksplorować odległe zakątki świata lub przeprowadzać skomplikowane eksperymenty laboratoryjne bez konieczności opuszczania sali lekcyjnej (4). Wirtualne symulacje i modele pozwalają na lepsze zrozumienie skomplikowanych konceptów i teorii poprzez praktyczne zastosowanie wiedzy.

Kluczowym elementem, który determinuje jakość i skuteczność doświadczenia VR, jest precyzyjne odwzorowanie ruchów użytkownika. To właśnie dokładne śledzenie i odwzorowanie ruchów pozwala na pełne zanurzenie w wirtualnym świecie i naturalne interakcje z otoczeniem. Realizm tych interakcji wpływa bezpośrednio na stopień immersji, czyli poczucie obecności w wirtualnym środowisku. W przypadku ruchu kończyn dolnych, precyzyjne odwzorowanie ruchów nóg jest szczególnie ważne dla aplikacji takich jak gry VR, treningi sportowe, czy rehabilitacja ruchowa.

Praca magisterska koncentruje się na stworzeniu systemu, który umożliwia odwzorowanie ruchu kończyn dolnych użytkownika w środowisku VR z wykorzystaniem czujników zamontowanych na smartfonach. Wybór smartfonów jako platformy dla czujników ruchu wynika z ich powszechnej dostępności, zaawansowanych technologii pomiarowych. Smartfony wyposażone są w różnego rodzaju czujniki, takie jak akcelerometry, żyroskopy i magnetometry, które mogą precyzyjnie monitorować ruchy i orientację urządzenia.

Projekt zakłada wykorzystanie tych czujników do zbierania danych o ruchach nóg użytkownika i przesyłania ich do systemu VR za pomocą protokołu UDP (User Datagram Protocol) (5). System ten składa się z dwóch aplikacji: pierwszej, działającej na smartfonach, która zbiera dane z czujników i przesyła je do drugiej aplikacji na komputerze. Druga aplikacja odbiera te dane i wykorzystuje je do animacji ruchu nóg awatara w wirtualnym środowisku. Taki układ pozwala na symulację naturalnych ruchów kończyn dolnych, co zwiększa realizm i immersję doświadczenia VR.

Realizacja tego projektu ma na celu nie tylko zbadanie technicznych aspektów wykorzystania czujników ruchu wbudowanych w smartfony do śledzenia ruchów kończyn dolnych, ale także ocenę praktycznych możliwości i ograniczeń tego rozwiązania w kontekście aplikacji VR. Wyniki badań mogą przyczynić się do dalszego rozwoju technologii VR, szczególnie w obszarze interakcji użytkownika z wirtualnym środowiskiem poprzez naturalne ruchy ciała.

Cel pracy

Celem pracy jest opracowanie systemu umożliwiającego odwzorowanie ruchów kończyn dolnych w środowisku wirtualnej rzeczywistości. System ten powinien zapewnić naturalne i intuicyjne odwzorowanie ruchów użytkownika na awatara w grze VR, co przyczyni się do zwiększenia realizmu i immersji doświadczenia wirtualnego. Ponadto, praca ma na celu zbadanie efektywności zastosowania wbudowanych w smartfony czujników ruchu do tego typu zadań oraz ocenić potencjalne problemy i wyzwania związane z ich wykorzystaniem.

Zakres projektu

Projekt obejmuje opracowanie dwóch aplikacji w środowisku Unity: jednej działającej na smartfonach, która wykorzystuje wbudowane czujniki do zbierania danych o ruchu i przesyłania ich za pomocą protokołu UDP przez WiFi, oraz drugiej działającej na komputerze, która odbiera te dane i odwzorowuje ruchy kończyn dolnych awatara w środowisku VR. Każda aplikacja na smartfonie jest odpowiedzialna za przesyłanie danych z czujnika a telefon musi być przytwierdzony na nogach użytkownika (udo lewe, kostka lewa, udo prawe, kostka prawa), co umożliwia pełne odwzorowanie ruchów kończyn dolnych. Warto wspomnieć, że aplikacja nie wymaga połączenia wszystkie czterech telefonów można je w różny sposób łączyć

(minimum jeden telefon maksymalnie cztery). Zakres projektu obejmuje również testowanie i walidację systemu, w celu oceny jego dokładności i użyteczności.

Struktura pracy

Praca została podzielona na rozdziały, które omawiają wszystkie aspekty projektu.

Rozdział Pierwszy prezentuje przegląd literatury związanej z wirtualną rzeczywistością i czujnikami ruchu, a także omawia dotychczasowe zastosowania tych technologii w nauce i rozrywce.

W rozdziale drugim przedstawiono założenia projektowe, architekturę systemu oraz opis użytego sprzętu i oprogramowania.

Rozdział trzeci zasady działania aplikacji wraz z przedstawieniem ich w formie schematu i diagramu.

Rozdział czwarty zawiera szczegółowy opis implementacji obu aplikacji, w tym sposób zbierania i przesyłania danych z czujników oraz integrację tych danych w aplikacji VR.

Rozdział piąty opisuje metody testowania systemu, prezentuje i analizuje wyniki testów.

W rozdziale szóstym przedstawiono podsumowanie projektu, omówiono osiągnięcia, zidentyfikowano ograniczenia i trudności napotkane podczas realizacji projektu oraz zaproponowano możliwości dalszego rozwoju systemu.

Pracę kończy bibliografia, spis rysunków, które ułatwiają odnalezienie i zrozumienie zawartych w pracy informacji.

Praca ma na celu nie tylko przedstawienie teoretycznych i praktycznych aspektów odwzorowania ruchu kończyn dolnych w środowisku VR, ale również dostarczenie szczegółowych instrukcji i kodu źródłowego, które mogą być wykorzystane przez innych badaczy i twórców aplikacji VR. Praca ta stanowi wkład w rozwój technologii VR, zwłaszcza w kontekście interakcji użytkownika z wirtualnym środowiskiem za pomocą naturalnych ruchów ciała.

Spis skrótów i oznaczeń

Lista najważniejszych skrótów i oznaczeń używanych w pracy:

1. **VR** - Virtual Reality (Wirtualna Rzeczywistość)
   - Technologia pozwalająca na stworzenie wirtualnego środowiska, w którym użytkownik może się zanurzyć i interaktywnie z nim oddziaływać.
2. **UDP** - User Datagram Protocol
   - Protokół komunikacyjny wykorzystywany do przesyłania danych w sieciach komputerowych, charakteryzujący się brakiem gwarancji dostarczenia, ale dużą szybkością transmisji.
3. **XYZ** - Osie współrzędnych w trójwymiarowej przestrzeni
   - Standardowy system współrzędnych stosowany do opisu położenia punktów w przestrzeni trójwymiarowej.
4. **Kwaterniony**
   - Matematyczna reprezentacja orientacji i rotacji w przestrzeni trójwymiarowej, używana do uniknięcia problemów z gimbal lock w obliczeniach rotacyjnych.
5. **IP** - Internet Protocol (Protokół internetowy)
   - Protokół odpowiedzialny za adresowanie i trasowanie pakietów danych w sieciach komputerowych.
6. **WiFi** - Wireless Fidelity
   - Technologia bezprzewodowej komunikacji w sieciach lokalnych, umożliwiająca urządzeniom komunikację bez użycia kabli.
7. **IDE** - Integrated Development Environment (Zintegrowane środowisko programistyczne)
   - Aplikacja zapewniająca wszechstronne narzędzia do pisania, testowania i debugowania kodu źródłowego.

Te skróty i oznaczenia będą używane w całej pracy, aby ułatwić opis technologii, metod i narzędzi używanych w projekcie.

1 Studium literaturowe dotyczące odwzorowania motoryki kończyn dolnych.

Studium literaturowe na temat odwzorowania motoryki kończyn dolnych obejmuje przegląd i analizę istniejących technologii oraz metod stosowanych w śledzeniu i odwzorowaniu ruchów kończyn dolnych w wirtualnej rzeczywistości (VR). W tej części pracy skupiono się na różnych typach czujników ruchu, technologiach fuzji sensorycznej oraz ich zastosowaniach w urządzeniach mobilnych i systemach VR. Ponadto, omówiono możliwe sposoby przesyłu danych, w tym charakterystykę protokołu UDP, jego porównanie z TCP oraz zastosowania w systemach czasu rzeczywistego. Dodatkowo, przeanalizowano kwaterniony, ich podstawy matematyczne, porównanie z kątami Eulera oraz zastosowanie w grafice komputerowej. Na koniec dokonano przeglądu istniejących rozwiązań z zakresu odwzorowania motoryki kończyn dolnych.

1.1 Czujniki ruchu i ich zastosowania

Czujniki ruchu odgrywają kluczową rolę w precyzyjnym śledzeniu i odwzorowaniu ruchów użytkownika, co jest niezbędne dla realistycznych i immersyjnych doświadczeń w wirtualnej rzeczywistości. Dzięki nim możliwe jest dokładne monitorowanie pozycji i orientacji ciała użytkownika, co pozwala na naturalną interakcję z wirtualnym środowiskiem.
W ramach studium literaturowego omówiono różne rodzaje czujników ruchu, technologie fuzji sensorycznej oraz ich zastosowania w urządzeniach mobilnych i systemach VR. Szczególną uwagę poświęcono czujnikowi Game Rotation Vector, który będzie wykorzystywany w projekcie do monitorowania ruchów kończyn dolnych użytkownika.

1.1.1 Rodzaje czujników ruchu

Czujniki ruchu w technologiach VR i AR dzielą się na kilka głównych kategorii, które są niezbędne do precyzyjnego śledzenia i interakcji użytkownika ze środowiskiem wirtualnym (6) (7) (8). Do najbardziej rozpowszechnionych należą:

- Akcelerometry: Pozwalają na pomiar przyspieszenia liniowego w różnych osiach. Używane są nie tylko do określania orientacji urządzenia (np. pionowej czy poziomej), ale także do wykrywania ruchu i szybkości przesunięcia (9).

- Żyroskopy: Wykorzystywane do mierzenia i utrzymania orientacji urządzenia poprzez detekcję rotacji wokół osi.
- Magnetometry: Umożliwiają określenie orientacji urządzenia względem pola magnetycznego Ziemi, co jest szczególnie przydatne w kompasach czy systemach lokalizacji.
- Kamery i sensory optyczne: Stosowane w zaawansowanych systemach VR do precyzyjnego śledzenia ruchów użytkownika i jego interakcji z otoczeniem. Technologie takie jak Time-of-Flight (ToF) i różne formy rozpoznawania struktury światła umożliwiają dokładne mapowanie przestrzeni i obiektów

1.1.2 Czujniki ruchu w urządzeniach mobilnych

Czujniki ruchu są szeroko stosowane w urządzeniach mobilnych, gdzie pełnią kluczową rolę w różnych aplikacjach, od prostych funkcji użytkowych po zaawansowane interakcje w rzeczywistości rozszerzonej (AR) i wirtualnej rzeczywistości (VR). Smartfony wyposażone są zazwyczaj w kombinację akcelerometrów, żyroskopów i magnetometrów, co umożliwia dokładne śledzenie ruchów i orientacji urządzenia.

Akcelerometry w smartfonach mierzą przyspieszenie liniowe, co pozwala na wykrywanie ruchów takich jak potrząsanie czy zmiana pozycji urządzenia. Żyroskopy dostarczają informacji o rotacji, co jest niezbędne do śledzenia bardziej złożonych ruchów. Magnetometry pozwalają na orientację względem pola magnetycznego Ziemi, co jest przydatne w aplikacjach nawigacyjnych. Integracja tych czujników umożliwia tworzenie aplikacji AR i VR, które reagują na ruchy użytkownika w czasie rzeczywistym, oferując bardziej immersyjne i interaktywne doświadczenia.

1.1.3 Czujnik Game Rotation Vector

Jednym z zaawansowanych czujników wykorzystywanych w smartfonach jest czujnik **Game Rotation Vector** opisany w dokumentacji androida (10). Jest to czujnik, który łączy dane z akcelerometru i żyroskopu, aby określić orientację urządzenia w przestrzeni. Działa na

zasadzie analizy rotacji w trzech osiach, dostarczając wartości odpowiadające obrotowi wokół osi X, Y i Z.

Czujnik Game Rotation Vector działa w następujący sposób:

1. **Zbieranie danych**: Akcelerometr dostarcza informacje o przyspieszeniu liniowym, a żyroskop mierzy prędkość kątową urządzenia.
2. **Fuzja czujników**: Dane z akcelerometru i żyroskopu są łączone przy użyciu algorytmu fuzji czujników. Akcelerometr jest dobry w wykrywaniu orientacji statycznej, ale podatny na zakłócenia w przypadku dynamicznych ruchów. Żyroskop jest bardzo precyzyjny w krótkim okresie czasu, ale jego dokładność może się zmniejszać z powodu dryftu. Algorytm fuzji czujników wykorzystuje zalety obu tych czujników, aby zapewnić dokładne i stabilne wyniki orientacji.
3. **Wykorzystanie kwaternionów**: Wynikiem fuzji jest zestaw kwaternionów, które opisują orientację urządzenia w przestrzeni trójwymiarowej. Kwaterniony pozwalają na uniknięcie problemów związanych z gimbal lock, zapewniając płynne i stabilne śledzenie rotacji.
4. **Wyjście danych**: Czujnik zwraca wartości, które są bezpośrednio wykorzystywane do odwzorowania orientacji w aplikacjach VR.

W projekcie zostaną wykorzystane czujniki Game Rotation Vector zamontowane w smartfonach, będą one używane do monitorowania ruchów kończyn dolnych użytkownika. Dane dotyczące rotacji nóg, zbierane przez te czujniki, będą przekazywane do systemu VR za pomocą protokołu UDP.

1.2 Protokół UDP i jego zastosowanie

W kontekście śledzenia ruchów w systemach rzeczywistości wirtualnej (VR) oraz rozszerzonej (AR), efektywna transmisja danych jest kluczowa. Protokół User Datagram Protocol (UDP) jest szeroko stosowany ze względu na swoją szybkość i niskie opóźnienia. W przeciwieństwie do bardziej skomplikowanego i niezawodnego Transmission Control Protocol (TCP), UDP umożliwia przesyłanie danych bez konieczności nawiązywania połączenia, co przyspiesza komunikację. Jest to szczególnie ważne w aplikacjach czasu rzeczywistego, gdzie priorytetem jest szybkość przesyłu danych, a nie ich niezawodność.

1.2.1 Charakterystyka UDP

User Datagram Protocol (UDP) jest protokołem komunikacyjnym warstwy transportowej, który jest częścią zestawu protokołów internetowych (IP). W przeciwieństwie do Transmission Control Protocol (TCP), UDP jest protokołem bezpołączeniowym, co oznacza, że nie ustanawia sesji komunikacyjnej między nadawcą a odbiorcą przed przesyłaniem danych (5). Dzięki temu UDP umożliwia szybkie przesyłanie danych, ale kosztem braku gwarancji ich dostarczenia, porządku i ochrony przed duplikacją. UDP jest prostym, lekkim protokołem, który nie oferuje mechanizmów korekcji błędów ani potwierdzeń odbioru, co oznacza, że aplikacje korzystające z UDP muszą być w stanie tolerować utratę pakietów, błędy i duplikacje. UDP jest często wykorzystywany w aplikacjach, które wymagają niskich opóźnień, takich jak strumieniowanie wideo, VoIP (Voice over IP) i gry online (11).

1.2.2 Porównanie UDP i TCP

TCP (Transmission Control Protocol) i UDP (User Datagram Protocol) są dwoma głównymi protokołami warstwy transportowej w sieciach IP, ale mają różne właściwości i zastosowania (5):

TCP:

- Połączeniowy: TCP ustanawia połączenie między nadawcą a odbiorcą przed przesyłaniem danych (tzw. handshake).
- Niezawodność: TCP zapewnia dostarczenie danych, ich porządek i ochronę przed duplikacją dzięki mechanizmom potwierdzania odbioru i retransmisji.
- Kontrola przepływu i przeciążenia: TCP dostosowuje szybkość przesyłania danych w zależności od warunków sieciowych, co zwiększa niezawodność, ale wprowadza dodatkowe opóźnienia.

UDP:

- Bezpołączeniowy: UDP nie ustanawia połączenia przed przesyłaniem danych, co umożliwia szybszą transmisję.

- Niezawodność: Brak mechanizmów potwierdzania odbioru i retransmisji, co oznacza, że dane mogą zostać utracone, zduplikowane lub przyjść w niewłaściwej kolejności.
- Brak kontroli przepływu i przeciążenia: UDP nie dostosowuje szybkości przesyłania danych, co może prowadzić do przeciążenia sieci, ale minimalizuje opóźnienia.

1.2.3 Zastosowania UDP w systemach czasu rzeczywistego

UDP jest szczególnie przydatny w aplikacjach, które wymagają niskich opóźnień i mogą tolerować utratę pakietów, takich jak:

- **Strumieniowanie multimediów**: Aplikacje do strumieniowania wideo i audio często korzystają z UDP, ponieważ opóźnienia są bardziej krytyczne niż sporadyczne utraty danych.
- **VoIP (Voice over IP)**: Połączenia głosowe przez internet korzystają z UDP, aby zapewnić płynność rozmów, nawet jeśli niektóre pakiety zostaną utracone.
- **Gry online**: Gry wymagają szybkiej i płynnej komunikacji, co sprawia, że UDP jest idealnym wyborem, pomimo potencjalnych utrat pakietów.
- **Systemy czasu rzeczywistego**: Wszelkie aplikacje wymagające szybkiej reakcji na dane, takie jak systemy monitoringu i kontroli, korzystają z UDP, aby minimalizować opóźnienia transmisji.

Te właściwości sprawiają, że UDP jest idealnym wyborem do zastosowań w systemach VR, gdzie niskie opóźnienia są kluczowe dla realistycznego odwzorowania ruchów użytkownika.

1.3 Kwaterniony i ich zastosowanie w grafice komputerowej

Kwaterniony są zaawansowaną matematyczną reprezentacją rotacji w przestrzeni trójwymiarowej, która oferuje wiele zalet w porównaniu z tradycyjnymi metodami, takimi jak kąty Eulera. Kwaterniony, wprowadzone przez Williama Hamiltona w 1843 roku, są czterowymiarowymi liczbami składającymi się z jednego składnika rzeczywistego i trzech składników urojonych. Umożliwiają one reprezentację rotacji bez ryzyka wystąpienia problemów z gimbal lock oraz zapewniają płynne i stabilne śledzenie orientacji obiektów (12).

Kwaterniony są powszechnie stosowane w grafice komputerowej, symulacjach, robotyce oraz nawigacji. Dzięki nim możliwe jest efektywne i precyzyjne śledzenie rotacji.

1.3.1 Podstawy matematyczne kwaternionów

Kwaterniony można przedstawić w postaci:

$$q = ae + bi + cj + dk$$

gdzie a,b,c,d są liczbami rzeczywistymi, a (i, j, k) są jednostkami urojonymi. Rotacje w przestrzeni trójwymiarowej mogą być opisane za pomocą jednostkowych kwaternionów, które mają normę równą 1. Jak można to zobaczyć na interaktywnej prezentacji (12). Rotacja jest reprezentowana przez kwaternion o postaci:

$$q = cos\left(\frac{\theta}{2}\right) + sin\left(\frac{\theta}{2}\right)(xi + yj + zk)$$

Gdzie θ jest kątem rotacji, a (x,y,z) jest jednostkowym wektorem osi rotacji. Podział przez 2 wynika z tego, że kwaterniony opisują połowę kąta rotacji, co jest istotne przy operacjach związanych z rotacją.

1.3.2 Kwaterniony vs kąty Eulera oraz gimbal lock

Kąty Eulera opisują rotację wokół trzech osi (X, Y, Z), jednakże mogą prowadzić do sytuacji, w której dwie z osi rotacji stają się współliniowe, co powoduje utratę jednego stopnia swobody (taki przypadek zwanym gimbal lock). Tego problemu można uniknąć, stosując kwaterniony, które opisują rotację w przestrzeni trójwymiarowej za pomocą czterech parametrów: q_0 (skalarny) oraz q_1, q_2, q_3 (wektorowych).

1.3.3 Zastosowanie kwaternionów w grafice komputerowej

W systemach VR i animacji, kwaterniony są używane do reprezentowania orientacji obiektów i awatarów w przestrzeni trójwymiarowej. Są one szczególnie przydatne w aplikacjach śledzenia ruchu, gdzie precyzja i stabilność są kluczowe. Kwaterniony pozwalają

na odwzorowanie skomplikowanych ruchów bez utraty dokładności, co jest istotne dla zapewnienia realistycznego doświadczenia użytkownika.

Główne zalety kwaternionów w grafice komputerowej i systemach śledzenia ruchu to:

1. Brak gimbal lock: Kwaterniony nie cierpią na problem gimbal lock, co zapewnia płynne i nieprzerwane śledzenie rotacji w każdej orientacji.
2. Interpolacja: Kwaterniony umożliwiają łatwą i efektywną interpolację rotacji, znaną jako slerp (spherical linear interpolation). Jest to kluczowe dla płynnego przejścia między różnymi orientacjami w animacji.
3. Kompaktowość: Kwaterniony są bardziej kompaktowe niż macierze rotacji, co czyni je bardziej efektywnymi w obliczeniach i przechowywaniu danych.
4. Stabilność obliczeń: Algorytmy oparte na kwaternionach są mniej podatne na błędy numeryczne, co zapewnia stabilność obliczeń przy wielokrotnych operacjach rotacyjnych. Użycie kwaternionów zapewnia płynność i realizm odwzorowania rotacji, co przyczynia się do zwiększenia immersji i jakości doświadczenia VR.

1.4 Przegląd istniejących rozwiązań z zakresu odwzorowania motoryki kończyn dolnych.

Odwzorowanie motoryki kończyn dolnych w systemach wirtualnej rzeczywistości (VR) jest kluczowym elementem zapewniającym realistyczne i immersyjne doświadczenie użytkownika. Istnieje wiele różnych podejść do śledzenia ruchów ciała, które różnią się pod względem technologii, dokładności i kosztów.

1.4.1 Driver4VR

Jednym z popularnych rozwiązań jest aplikacja **Driver4VR**, która oferuje darmowe śledzenie całego ciała (Full Body Tracking) za pomocą odczytów z kamery (13). Aplikacja ta pozwala na odwzorowanie ruchów ciała użytkownika bez konieczności używania drogich czujników czy specjalistycznego sprzętu. Użytkownik może korzystać z kamer internetowych lub kamery Kinect, aby śledzić ruchy ciała i przenieść je do środowiska VR (14).

Driver4VR działa w następujący sposób:

1. **Instalacja i konfiguracja**: Użytkownik instaluje aplikację na swoim urządzeniu oraz konfiguruje kamerę, która będzie używana do śledzenia ruchów.
2. **Śledzenie ruchów**: Kamera rejestruje ruchy użytkownika, które następnie są przetwarzane przez algorytmy aplikacji w celu określenia pozycji i orientacji kończyn dolnych oraz innych części ciała.
3. **Integracja z VR**: Dane o ruchach ciała są przekazywane do systemu VR, który odwzorowuje je na awatarze użytkownika, zapewniając realistyczne odwzorowanie ruchów w wirtualnym środowisku.

Aplikacja ta jest dostępna na platformie Google Play i może być używana z różnymi zestawami VR, co czyni ją wszechstronnym narzędziem dla entuzjastów VR, którzy chcą poprawić swoje doświadczenia bez dużych inwestycji finansowych

Wady rozwiązania Driver4VR:

- **Zależność od kamery**: Skuteczność śledzenia ruchów zależy od jakości i ustawienia kamery, co może być problematyczne w warunkach słabego oświetlenia.

- **Opóźnienia i dokładność**: Odczyty z kamery mogą być mniej dokładne i mieć większe opóźnienia w porównaniu do specjalistycznych czujników ruchu.
- **Gubienie rotacji**: kamera nie zawsze wykrywa czy osoba stoi przodem czy tyłem.

Zalety rozwiązania projektowego z wykorzystaniem 4 smartfonów:

- **Niezależność od warunków oświetleniowych**: Czujniki ruchu w smartfonach (akcelerometry, żyroskopy, magnetometry) działają niezależnie od warunków oświetleniowych, co zapewnia większą niezawodność i dokładność w różnych środowiskach.
- **Precyzyjne śledzenie ruchów**: Smartfony wyposażone w czujniki Game Rotation Vector mogą precyzyjnie śledzić ruchy kończyn dolnych w czasie rzeczywistym, co zapewnia płynne i realistyczne odwzorowanie w środowisku VR

Podsumowując, Driver4VR jest efektywnym narzędziem do śledzenia ruchów ciała w VR dla użytkowników poszukujących nisko kosztowych rozwiązań. Jednakże, ograniczenia związane z użyciem kamery mogą wpływać na dokładność i jakość śledzenia. Rozwiązanie projektowe, wykorzystujące smartfony, oferuje bardziej niezawodną i precyzyjną alternatywę dla śledzenia ruchów kończyn dolnych w VR.

1.4.2 Vive Trackers

Vive Trackers to zaawansowane urządzenia śledzące opracowane przez firmę HTC, które umożliwiają precyzyjne odwzorowanie ruchów ciała użytkownika w środowisku VR. Trackery te są kompatybilne z systemem HTC Vive i oferują szeroką gamę zastosowań, od gier VR po profesjonalne aplikacje treningowe i symulacyjne (15).

Działanie Vive Trackers:

1. **Instalacja i konfiguracja**: Vive Trackers są łatwe do zamontowania na różnych częściach ciała lub obiektach. Po zainstalowaniu wymagają one konfiguracji w systemie VR, aby zsynchronizować ich położenie i orientację z wirtualnym środowiskiem.
2. **Śledzenie ruchów**: Trackery wykorzystują technologię śledzenia na podczerwień oraz system stacji bazowych, które emitują sygnały, odbierane przez sensory na trackerach. Dzięki temu możliwe jest precyzyjne śledzenie pozycji i orientacji trackerów w czasie rzeczywistym.

3. **Integracja z VR**: Dane z trackerów są przesyłane do systemu VR, gdzie są przetwarzane i wykorzystywane do odwzorowania ruchów użytkownika na awatarze w wirtualnym środowisku. To zapewnia realistyczne i responsywne interakcje w VR.

Zalety Vive Trackers:

- **Wysoka precyzja**: Dzięki zaawansowanej technologii śledzenia na podczerwień, Vive Trackers oferują bardzo wysoką precyzję śledzenia ruchów, co przekłada się na realistyczne odwzorowanie w VR.
- **Elastyczność zastosowań**: Trackery mogą być zamontowane na różnych częściach ciała, jak również na obiektach, co pozwala na szeroką gamę zastosowań, od gier po profesjonalne treningi i symulacje.
- **Integracja z ekosystemem HTC Vive**: Vive Trackers są w pełni kompatybilne z systemem HTC Vive, co umożliwia łatwą integrację z istniejącymi rozwiązaniami VR.

Wady Vive Trackers:

- **Koszt**: Vive Trackers są stosunkowo drogie, co może stanowić barierę dla indywidualnych użytkowników. Koszt samego trackera oraz dodatkowych stacji bazowych może znacznie zwiększyć całkowity wydatek.
- **Zależność od ekosystemu HTC**: Trackery są zaprojektowane do współpracy z systemem HTC Vive, co ogranicza ich kompatybilność z innymi systemami VR.

Porównanie z projektem:

- **Koszt**: Rozwiązanie opracowane w tym projekcie opiera się na wykorzystaniu powszechnie dostępnych smartfonów, co znacznie obniża koszty w porównaniu do dedykowanych trackerów takich jak Vive Trackers.
- **Niezależność od specyficznych systemów**: Rozwiązanie wykorzystujące smartofny może być bardziej uniwersalne i nie jest ograniczone do ekosystemu jednego producenta, co zwiększa jego dostępność dla szerokiej gamy użytkowników.
- **Elastyczność i dostępność**: Smartfony wyposażone w czujniki ruchu (akcelerometry, żyroskopy, magnetometry) są szeroko dostępne i mogą być łatwo konfigurowane oraz integrowane z różnymi systemami VR.

Podsumowując, chociaż Vive Trackers oferują wysoką precyzję i elastyczność zastosowań, ich wysoki koszt i zależność od ekosystemu HTC mogą być ograniczeniami dla wielu użytkowników. Rozwiązanie z wykorzystaniem smartfonów oferuje bardziej przystępną cenowo i uniwersalną alternatywę dla odwzorowania ruchów kończyn dolnych w VR.

1.4.3 Full Body Tracking SDK(Oculus)

Full Body Tracking SDK opracowany przez Oculus jest zaawansowanym rozwiązaniem do śledzenia ruchów ciała użytkownika za pomocą kamer wbudowanych w urządzenia Oculus. SDK to umożliwia integrację funkcji śledzenia ciała z aplikacjami VR, zapewniając realistyczne odwzorowanie ruchów górnych partii ciała w wirtualnym środowisku (16).

Działanie Full Body Tracking SDK:

1. **Instalacja i konfiguracja**: Użytkownicy muszą zainstalować odpowiednie oprogramowanie SDK i skonfigurować kamery w urządzeniu Oculus, które będą wykorzystywane do śledzenia ruchów.
2. **Śledzenie ruchów**: Kamery rejestrują ruchy ciała użytkownika, a algorytmy SDK przetwarzają te dane, aby określić pozycje i orientacje różnych części ciała.
3. **Integracja z VR**: Dane śledzenia są przekazywane do aplikacji VR, która odwzorowuje ruchy użytkownika na awatarze w wirtualnym środowisku, zapewniając interaktywne i immersyjne doświadczenie.

Zalety Full Body Tracking SDK:

- **Precyzyjne śledzenie**: Dzięki zaawansowanym algorytmom przetwarzania obrazu, SDK oferuje precyzyjne śledzenie ruchów górnych partii ciała, co może znacząco poprawić realizm doświadczeń VR.
- **Łatwa integracja**: SDK jest zaprojektowane do łatwej integracji z aplikacjami VR opartymi na platformie Oculus, co ułatwia deweloperom wprowadzenie funkcji śledzenia ciała.

Wady Full Body Tracking SDK:

- **Ograniczenia w śledzeniu nóg**: Chociaż SDK dobrze radzi sobie ze śledzeniem górnych partii ciała, nie jest zalecane do śledzenia ruchów nóg. Kamery umieszczone w urządzeniu Oculus mają ograniczone pole widzenia, co utrudnia dokładne śledzenie dolnych partii ciała.
- **Zależność od sprzętu Oculus**: SDK jest zaprojektowane do pracy wyłącznie z urządzeniami Oculus, co ogranicza jego kompatybilność z innymi systemami VR.

Porównanie z rozwiązaniem zaproponowanym w pracy:

- **Śledzenie nóg**: Rozwiązanie z wykorzystaniem do czterech smartfonów, z czujnikami Game Rotation Vector, jest zaprojektowane specjalnie do precyzyjnego śledzenia ruchów nóg, co stanowi istotną przewagę nad Full Body Tracking SDK, które nie jest zalecane do tego celu.
- **Uniwersalność i koszty**: Wykorzystanie powszechnie dostępnych smartfonów sprawia, że rozwiązanie zaproponowane w projekcie jest bardziej przystępne cenowo i uniwersalne w porównaniu do dedykowanego sprzętu Oculus. Smartfony mogą być łatwo konfigurowane i integrowane z różnymi systemami VR, podczas gdy SDK Oculus jest ograniczone do ekosystemu Oculus.

Podsumowując, chociaż Full Body Tracking SDK od Oculus oferuje zaawansowane funkcje śledzenia ruchów górnych partii ciała, jego ograniczenia w zakresie śledzenia nóg i zależność od sprzętu Oculus mogą być istotnymi wadami. Rozwiązanie bazujące na czujnikach w smartfonach, oferuje bardziej uniwersalną i przystępną cenowo alternatywę, która zapewnia precyzyjne śledzenie ruchów nóg w VR.

1.4.4 SlimeVR

SlimeVR to projekt, który oferuje rozwiązania do śledzenia ruchów ciała w VR, głównie oparte na czujnikach IMU (Inertial Measurement Unit). Projekt ten jest znany z dostarczania szczegółowych instrukcji, jak samodzielnie stworzyć trackery, a także prowadzi sklep z komponentami potrzebnymi do ich budowy (17).

Działanie SlimeVR:

1. **Instalacja i konfiguracja**: Użytkownicy mogą samodzielnie zbudować trackery IMU, korzystając z instrukcji dostępnych na stronie projektu. Po złożeniu trackerów, użytkownicy konfigurują je, aby komunikowały się z systemem VR.
2. **Śledzenie ruchów**: Trackery IMU rejestrują ruchy użytkownika i przesyłają dane do systemu VR. IMU trackery wykorzystują akcelerometry, żyroskopy i magnetometry do dokładnego śledzenia pozycji i orientacji.
3. **Integracja z VR**: Dane śledzenia są przesyłane do systemu VR, gdzie są przetwarzane i używane do odwzorowania ruchów użytkownika na awatarze w wirtualnym środowisku. SlimeVR pozwala również na wykorzystanie smartfonów jako czujników, ale wymaga to użycia aplikacji od innych twórców. Jak opisują to na stronie GitHub takie podejście ogranicza funkcjonalność i kompatybilność (18).

Zalety SlimeVR:

- **Elastyczność i personalizacja**: Możliwość samodzielnego zbudowania trackerów pozwala na dostosowanie rozwiązania do indywidualnych potrzeb użytkownika.
- **Kompatybilność z różnymi systemami VR**: Trackery IMU mogą być zintegrowane z wieloma platformami VR, co czyni SlimeVR uniwersalnym rozwiązaniem.

Wady SlimeVR:

- **Złożoność budowy**: Dla niektórych użytkowników samodzielne zbudowanie i skonfigurowanie trackerów może być wyzwaniem, wymagającym wiedzy technicznej i umiejętności manualnych.
- **Brak natywnej aplikacji na smartfony**: Chociaż SlimeVR pozwala na wykorzystanie smartfonów jako czujników, projekt nie oferuje własnej aplikacji mobilnej.

Użytkownicy muszą korzystać z aplikacji innych twórców, co może być mniej wygodne i zintegrowane.

Porównanie z rozwiązaniem zaproponowanym w pracy:

- **Łatwość użycia**: Rozwiązanie projektowe oferuje dwie natywne aplikacje (jedna na smartfony, druga na komputer), co ułatwia konfigurację i integrację systemu. Użytkownicy nie muszą szukać zewnętrznych aplikacji ani tworzyć własnych trackerów.
- **Natywna aplikacja na smartfony**: Praca zawiera aplikację mobilną, co eliminuje potrzebę korzystania z zewnętrznych aplikacji do zbierania danych z czujników, to sprawia, że jest bardziej spójne i przyjazne dla użytkownika.
- **Uniwersalność i przystępność**: Wykorzystanie powszechnie dostępnych smartfonów z czujnikami ruchu (akcelerometry, żyroskopy, magnetometry) sprawia, że rozwiązanie zapoponowane w porojekcie jest bardziej dostępne i przystępne cenowo.

Podsumowując, chociaż SlimeVR oferuje elastyczne i personalizowane rozwiązania do śledzenia ruchów ciała w VR, jego złożoność budowy i brak natywnej aplikacji na smartfony mogą stanowić wyzwanie dla niektórych użytkowników. Podejście z natywnymi aplikacjami na smartfony i komputery, oferuje bardziej spójną, łatwiejszą do użycia i przystępną alternatywę do śledzenia ruchów kończyn dolnych w VR.

1.4.5 April-Tag VR Full Body Tracker

April-Tag VR Full Body Tracker to innowacyjne rozwiązanie, które wykorzystuje technologię śledzenia kodów QR (AprilTags) do odwzorowania ruchów kończyn dolnych w wirtualnej rzeczywistości. System ten jest oparty na kamerach, które śledzą specjalne kody QR przymocowane do ciała użytkownika, umożliwiając precyzyjne odwzorowanie ruchów w VR (19).

Działanie April-Tag VR Full Body Tracker:

1. **Instalacja i konfiguracja**: Użytkownik drukuje i przymocowuje kody QR (AprilTags) do różnych części ciała, w tym kończyn dolnych. Następnie konfiguruje kamerę, która będzie śledzić te kody.
2. **Śledzenie ruchów**: Kamera rejestruje pozycję i orientację kodów QR, a specjalne algorytmy przetwarzają te dane, aby określić ruchy i orientację kończyn użytkownika.
3. **Integracja z VR**: Dane z kamer są przesyłane do systemu VR, gdzie są używane do odwzorowania ruchów użytkownika na awatarze w wirtualnym środowisku, zapewniając dokładne i realistyczne odwzorowanie.

Zalety April-Tag VR Full Body Tracker:

- **Wysoka precyzja**: Śledzenie kodów QR umożliwia bardzo dokładne odwzorowanie ruchów ciała, co jest kluczowe dla realistycznych doświadczeń VR.
- **Łatwość implementacji**: Użytkownik może samodzielnie wydrukować kody QR i przymocować je do ciała, co sprawia, że rozwiązanie jest stosunkowo proste do wdrożenia.

Wady April-Tag VR Full Body Tracker:

- **Zależność od linii widzenia**: System wymaga, aby kody QR były zawsze widoczne dla kamery. Ograniczenia przestrzenne lub przeszkody mogą wpływać na skuteczność śledzenia.
- **Kompleksowa konfiguracja**: Konieczność drukowania i przymocowania kodów QR do ciała może być czasochłonna i mniej wygodna w porównaniu do innych metod śledzenia.

Porównanie z rozwiązaniem zaproponowanym w pracy:

- **Niezależność od linii widzenia**: Rozwiązanie z wykorzystaniem smartfonów z czujnikami Game Rotation Vector nie wymaga ciągłego widoku kamery, co eliminuje problemy związane z linią widzenia.
- **Łatwiejsza konfiguracja**: Użycie smartfonów eliminuje potrzebę drukowania i przymocowywania kodów QR, co upraszcza proces konfiguracji i zwiększa wygodę użytkowania.

- **Uniwersalność i dostępność**: Rozwiązanie projektowe może być stosowane w różnych środowiskach bez konieczności dodatkowych akcesoriów, co czyni je bardziej uniwersalnym i dostępnym dla szerokiej gamy użytkowników.

Podsumowując, chociaż April-Tag VR Full Body Tracker oferuje wysoką precyzję śledzenia, jego zależność od linii widzenia i skomplikowana konfiguracja mogą być ograniczeniami dla niektórych użytkowników. Rozwiązanie oparte na smartfonach z wbudowanymi czujnikami ruchu, oferuje bardziej uniwersalną i łatwiejszą alternatywę dla precyzyjnego śledzenia ruchów kończyn dolnych w VR.

2 Założenia projektowe

W tym rozdziale przedstawiono: opis ogólny i założenia projektowe systemu śledzenia ruchów kończyn dolnych w wirtualnej rzeczywistości (VR). Celem projektu jest opracowanie systemu, który umożliwi precyzyjne odwzorowanie ruchów kończyn dolnych użytkownika w środowisku VR przy użyciu czujników ruchu zamontowanych na smartfonach. Wprowadzone zostaną wymagania funkcjonalne i niefunkcjonalne, użyte technologie i narzędzia oraz kryteria sukcesu projektu.

2.1 Opis ogólny

Projekt ma na celu opracowanie systemu, który umożliwi precyzyjne odwzorowanie ruchów kończyn dolnych użytkownika w środowisku wirtualnej rzeczywistości (VR). System ten będzie składał się z dwóch aplikacji stworzonych w Unity: aplikacja mobilna będzie zbierała dane z czujników ruchu zamontowanych na smartfonach, a druga aplikacja komputerowa będzie odbierała te dane i odwzorowywała ruchy użytkownika na awatarze w VR.

Smartfony będą wyposażone w czujniki Game Rotation Vector, które zbierają dane o orientacji i ruchach kończyn dolnych użytkownika. Dane te będą przesyłane do aplikacji komputerowej za pomocą protokołu UDP, co zapewni niskie opóźnienia i wysoką precyzję odwzorowania ruchów w wirtualnym środowisku.

2.2 Wymagania funkcjonalne

Wymagania funkcjonalne określają, jakie funkcje i zadania musi spełniać system, aby osiągnąć zamierzony cel:

- **Dokładne odwzorowanie ruchu**: System musi zapewniać precyzyjne śledzenie i odwzorowanie ruchów kończyn dolnych użytkownika w czasie rzeczywistym.
- **Niskie opóźnienia**: Użycie protokołu UDP ma na celu zminimalizowanie opóźnień w przesyłaniu danych z czujników ruchu do aplikacji VR.
- **Stabilność i płynność animacji**: Użycie quaternionów do reprezentacji orientacji zapewni płynne i stabilne śledzenie rotacji, unikając problemów związanych z gimbal lock.
- **Łatwość użytkowania**: Aplikacje mobilne muszą być łatwe do zainstalowania i używania na smartfonach, a aplikacja komputerowa powinna być intuicyjna i łatwa do integracji z istniejącymi systemami VR.

2.3 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają ogólne cechy i parametry systemu, które nie są bezpośrednio związane z jego funkcjonalnością:

- **Skalowalność**: System powinien być łatwo rozbudowywany i skalowalny, aby umożliwić przyszłe rozszerzenia i integracje z innymi urządzeniami i technologiami VR.
- **Niezawodność**: System musi być niezawodny i odporny na utratę danych, z możliwością łatwej rekonfiguracji w przypadku awarii.
- **Wydajność**: System powinien działać wydajnie, zapewniając płynne i responsywne doświadczenia VR bez opóźnień i zakłóceń.
- **Kompatybilność**: Aplikacje powinny być kompatybilne z szeroką gamą urządzeń mobilnych oraz z różnymi środowiskami VR na komputerze.

2.4 Technologie i narzędzia

Do realizacji projektu zostaną wykorzystane następujące technologie i narzędzia:

2.4.1 Visual Studio 2022

Visual Studio 2022 to zaawansowane zintegrowane środowisko programistyczne (IDE) firmy Microsoft, umożliwiające tworzenie aplikacji w różnych językach programowania, takich jak C#, F#, C++, oraz umożliwiające tworzenie aplikacji na różne platformy, w tym .NET, web, mobile i cloud.

2.4.2 Język C#

C# to język programowania opracowany przez Microsoft, który jest powszechnie używany w tworzeniu aplikacji na platformę .NET. Język ten oferuje wysoką wydajność, bezpieczeństwo oraz szerokie wsparcie dla obiektowo zorientowanego programowania, co czyni go idealnym wyborem do tworzenia aplikacji VR w Unity.

2.4.3 Silnik Unity

Unity to zaawansowany silnik do tworzenia gier i aplikacji VR. Umożliwia tworzenie interaktywnych środowisk 3D, oferując szeroki zestaw narzędzi do modelowania, animacji i symulacji. Unity jest kompatybilne z różnymi platformami VR, co pozwala na łatwą integrację i szerokie zastosowanie w różnych środowiskach.

2.4.4 Xr Interaction Toolkit

XR Interaction Toolkit to narzędzie w Unity, które umożliwia tworzenie interaktywnych doświadczeń VR i AR. Toolkit ten zawiera gotowe komponenty i szablony do tworzenia interakcji w wirtualnej rzeczywistości, co znacznie przyspiesza proces tworzenia aplikacji VR.

2.4.5 GitHub

GitHub to platforma do zarządzania kodem źródłowym i współpracy zespołowej. Umożliwia wersjonowanie kodu, śledzenie zmian i zarządzanie projektami programistycznymi. Dzięki GitHub, można łatwo dzielić się kodem i monitorować postępy projektu.

2.5 Kryteria sukcesu projektu

Kryteria sukcesu projektu określają, jakie warunki muszą zostać spełnione, aby projekt został uznany za udany:

- **Dokładność odwzorowania ruchów**: Ruchy kończyn dolnych użytkownika są precyzyjnie odwzorowane na ruchy awatara w środowisku VR, z minimalnymi opóźnieniami.
- **Stabilność systemu**: System działa niezawodnie, bez częstych awarii i przestojów, a dane są przesyłane w sposób ciągły i bez zakłóceń.
- **Pozytywne testy użytkowników**: Testy z udziałem użytkowników końcowych potwierdzają, że system jest łatwy w użyciu i znacząco poprawia immersję i realizm doświadczenia VR.
- **Skalowalność i możliwość rozbudowy**: System jest zaprojektowany w sposób umożliwiający łatwą rozbudowę i integrację z dodatkowymi urządzeniami i technologiami.

Podsumowując, założenia projektowe stanowią fundament dla realizacji systemu odwzorowania ruchów kończyn dolnych w środowisku VR, zapewniając klarowne wytyczne dotyczące funkcjonalności, technologii i harmonogramu projektu.

3 Zasady działania aplikacji projektowej

W tej sekcji opisano zasady działania aplikacji projektowej, której celem jest odwzorowanie ruchów kończyn dolnych użytkownika w środowisku wirtualnej rzeczywistości (VR). Aplikacja składa się z dwóch głównych komponentów: aplikacji mobilnej działającej na smartfonach oraz aplikacji komputerowej. Komunikacja między tymi komponentami odbywa się za pośrednictwem protokołu UDP przez sieć WiFi.

3.1 Komponenty systemu

Aplikacja mobilna:

- **Funkcjonalność:** Aplikacja mobilna działa na smartfonach zamocowanych na nogach użytkownika i zbiera dane z czujników takich jak akcelerometr, żyroskop i Game Rotation Vector.

- **Przesył danych:** Dane są przesyłane w czasie rzeczywistym na odpowiednie porty komputera. Każdy smartfon przesyła dane na dedykowany port:
    - Udo lewe: Port 12345
    - Kostka lewa: Port 12346
    - Udo prawe: Port 12347
    - Kostka prawa: Port 12348

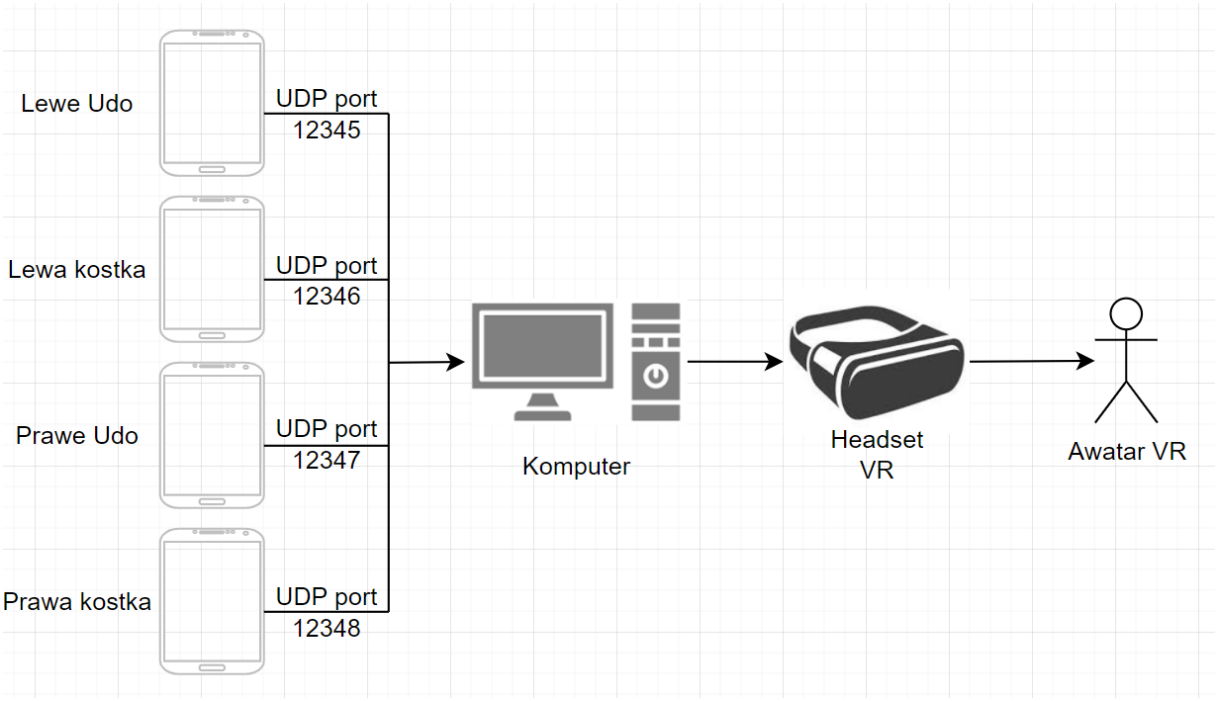
Aplikacja komputerowa:

- **Funkcjonalność:** Aplikacja komputerowa odbiera dane przesyłane z aplikacji mobilnej, deserializuje je i przetwarza. Dane są następnie używane do aktualizacji rotacji i pozycji odpowiednich części ciała awatara w środowisku VR.
- **Deserializacja:** Dane odbierane są w formacie tekstowym i przetwarzane za pomocą wyrażeń regularnych w celu uzyskania odpowiednich typów danych (Vector3 dla akcelerometru i żyroskopu, Quaternion dla orientacji, DateTime dla znacznika czasu).
- **Przetwarzanie:** Dane są przetwarzane i korygowane w czasie rzeczywistym, aby zapewnić płynne i dokładne odwzorowanie ruchów.

3.2 Schemat działania

Schemat działania aplikacji projektowej:

Schemat działania aplikacji projektowej został przedstawiony na rysunku (Rysunek 1).



Rysunek 1 Sposób planowanej komunikacji między czujnikami a systemem VR

Opis schematu:

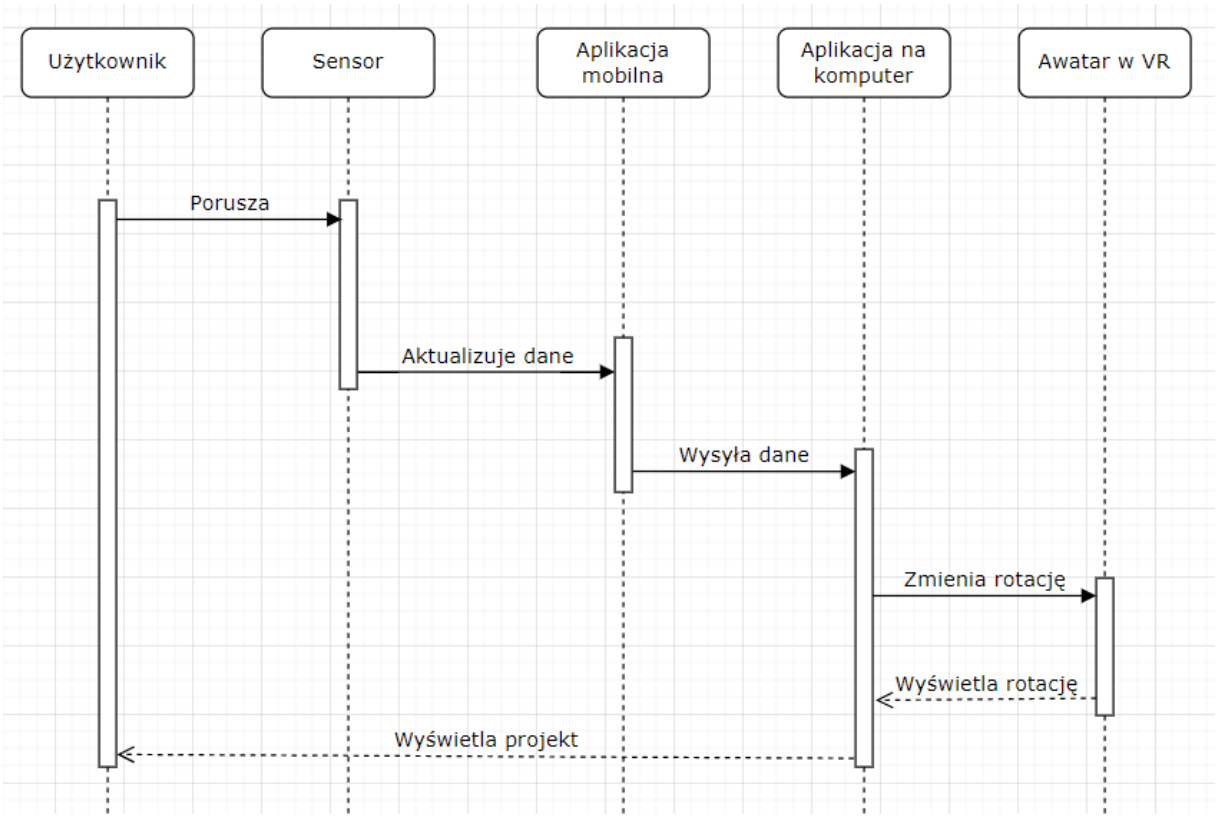
1. **Zbieranie danych:** Smartfony zbierają dane z czujników akcelerometru, żyroskopu i Game Rotation Vector.
2. **Przesył danych:** Dane są przesyłane poprzez UDP na odpowiednie porty komputera.
3. **Odbiór danych:** Komputer odbiera dane z różnych portów, deserializuje je i przetwarza.
4. **Wyświetlenie aplikacji w VR:** Komputer wyświetla projekt w przestrzeni VR.
5. **Aktualizacja awatara:** Przetworzone dane są używane do aktualizacji pozycji i rotacji odpowiednich części ciała awatara w środowisku VR.

Wnioski: Aplikacja projektowa pozwala na realistyczne odwzorowanie ruchów kończyn dolnych użytkownika w środowisku VR dzięki precyzyjnemu zbieraniu, przesyłaniu i

przetwarzaniu danych z czujników zamontowanych na smartfonach. System jest skalowalny i może działać z różną liczbą czujników, co umożliwia elastyczne dostosowanie do potrzeb użytkownika.

3.3 Diagram Sekwencji UML

Na rysunku (Rysunek2) przedstawiono diagram sekwencji UML dla aplikacji projektowej. Diagram ilustruje interakcje między użytkownikiem, czujnikami, aplikacją mobilną, aplikacją komputerową oraz awatarem w środowisku wirtualnej rzeczywistości (VR). Poniżej opisano szczegółowe etapy i działania poszczególnych komponentów systemu, które są ukazane na diagramie.



Rysunek 2 Diagram sekwencji UML

Diagram sekwencji przedstawia następujące kroki:

1. **Użytkownik porusza Sensor:**
    - Użytkownik wykonuje ruch, który jest rejestrowany przez sensor zamontowany na smartfonie. Sensor ten może być przytwierdzony do różnych części nogi (udo, kostka), co pozwala na precyzyjne śledzenie ruchów.
2. **Sensor aktualizuje dane aplikacji mobilnej:**
    - Aplikacja pobiera aktualne dane z sensora. Aplikacja mobilna zbiera dane z różnych czujników (akcelerometr, żyroskop, Game Rotation Vector) i przygotowuje je do przesłania.
3. **Aplikacja mobilna wysyła dane aplikacji na komputer:**
    - Aplikacja mobilna wysyła zebrane dane poprzez protokół UDP do aplikacji komputerowej. Dane te są przesyłane na odpowiednie porty w zależności od umiejscowienia sensora.
4. **Aplikacja na komputer zmienia rotację awatara:**
    - Aplikacja komputerowa odbiera dane z różnych portów, deserializuje je i przetwarza. Następnie na podstawie otrzymanych danych aktualizuje rotację awatara w środowisku VR, odzwierciedlając ruchy użytkownika.
5. **Awatar wyświetla rotację:**
    - Awatar w środowisku VR odzwierciedla ruchy użytkownika w czasie rzeczywistym. Każda zmiana pozycji i rotacji jest bezpośrednio przenoszona na awatara.
6. **Aplikacja na komputer wyświetla projekt:**
    - Aplikacja komputerowa może również wyświetlać dodatkowe informacje użytkownikowi, takie jak status połączenia. Użytkownik może monitorować i kontrolować proces odwzorowania ruchów.

7. **Użytkownik widzi efekt:**

   - Użytkownik obserwuje zmiany w VR, które są rezultatem jego ruchów. Użytkownik może doświadczać realistycznych interakcji w wirtualnym środowisku.

3.4 Zalety systemu

- **Elastyczność:** System może działać z jednym lub czterema smartfonami, co pozwala na dostosowanie poziomu precyzji śledzenia ruchów.
- **Realistyczne odwzorowanie:** Precyzyjne przetwarzanie danych z czujników zapewnia realistyczne odwzorowanie ruchów w środowisku VR.
- **Skalowalność:** Możliwość rozszerzenia funkcjonalności systemu o dodatkowe czujniki na innych częściach ciała w przyszłości.

3.5 Wyzwania i ograniczenia

- **Opóźnienia w przesyle danych:** Pomimo zwiększenia dozwolonego opóźnienia, nadal mogą występować pewne opóźnienia wpływające na płynność odwzorowania ruchów.
- **Mocowanie czujników:** Dokładność odwzorowania ruchów zależy od prawidłowego mocowania smartfonów na nogach użytkownika.

4 Implementacja

Implementacja projektu obejmuje stworzenie dwóch aplikacji w środowisku Unity: jednej na smartfony, która zbiera dane z czujników ruchu, oraz drugiej na komputer, która odwzorowuje ruchy użytkownika na awatarze w wirtualnej rzeczywistości (VR). Aplikacje te komunikują się ze sobą za pomocą protokołu UDP, aby zapewnić szybkie i niezawodne przesyłanie danych.

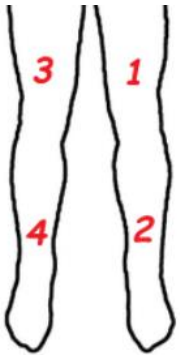
4.1 Aplikacja na telefony

Aplikacja mobilna będzie odpowiedzialna za zbieranie danych z czujników ruchu zamontowanych w smartfonach. Czujniki te obejmują akcelerometr, żyroskop oraz czujnik Game Rotation Vector, które pozwalają na precyzyjne śledzenie orientacji i ruchu urządzenia w przestrzeni.

4.1.1 Opis funkcjonalności

Aplikacja mobilna będzie miała następujące funkcje:

- **Zbieranie danych z czujników**: Aplikacja będzie zbierać dane z akcelerometru, żyroskopu oraz czujnika Game Rotation Vector w czasie rzeczywistym.
- **Przesyłanie danych**: Dane z czujników będą przesyłane za pomocą protokołu UDP do aplikacji komputerowej, przy użyciu odpowiednich portów dla każdej z kończyn, tak jak jest to narysowane na obrazku (Rysunek 3) (12345 - udo lewe, 12346 - kostka lewa, 12347 - udo prawe, 12348 - kostka prawa). Warto dodać, że aplikacja na komputer będzie też dostosowana do przypadków użycia mniej niż czterech telefonów np. telefon oznaczony na rysunku 1 i 2 same będą w stanie symulować nogę bez potrzeby połączenia 3 i 4 telefonu.



Rysunek 3 Umiejscowienie telefonów na nogach.

4.1.2 Inicjalizacja i odczyt z czujników

Dzięki wykorzystaniu Unity i jego InputSystem, możemy w łatwy sposób uzyskać dostęp do czujników pozycyjnych oraz rotacyjnych w skrypcie. Poniższy kod przedstawia, jak włączyć czujniki i uzyskać dostęp do ich pomiarów. W metodzie Start() aktywujemy czujniki, a w metodzie FixedUpdate(), która odpowiada za samoistne odświeżanie w Unity, pobieramy aktualne dane z czujników.

```
private void Start()
{
    Screen.sleepTimeout = SleepTimeout.NeverSleep;
    InputSystem.EnableDevice(Gyroscope.current);
    InputSystem.EnableDevice(Accelerometer.current);
    InputSystem.EnableDevice(AttitudeSensor.current);
}
private void FixedUpdate()
{
    InputSystem.Update();
    angularVelocity = Gyroscope.current.angularVelocity.ReadValue();
    acceleration = Accelerometer.current.acceleration.ReadValue();
    attitudeSensor = AttitudeSensor.current.attitude.ReadValue();
}
```

Wyjaśnienie kodu:

- **Metoda Start()**:
  - Screen.sleepTimeout = SleepTimeout.NeverSleep; - Zapobiega przejściu urządzenia w tryb uśpienia podczas działania aplikacji.
  - InputSystem.EnableDevice(Gyroscope.current); - Aktywuje żyroskop.
  - InputSystem.EnableDevice(Accelerometer.current); - Aktywuje akcelerometr.
  - InputSystem.EnableDevice(AttitudeSensor.current); - Aktywuje czujnik game rotation wektor.
- **Metoda FixedUpdate()**:
  - InputSystem.Update(); - Aktualizuje stan urządzeń wejściowych.
  - angularVelocity = Gyroscope.current.angularVelocity.ReadValue(); - Pobiera aktualne dane z żyroskopu.
  - acceleration = Accelerometer.current.acceleration.ReadValue(); - Pobiera aktualne dane z akcelerometru.
  - attitudeSensor = AttitudeSensor.current.attitude.ReadValue(); - Pobiera aktualne dane z czujnika game rotation wektor.

Aplikacja wymaga jedynie czujnika orientacji (AttitudeSensor, czyli Game Rotation Vector Sensor) do poprawnego działania. Pozostałe czujniki (żyroskop i akcelerometr) są również aktywowane i dane z nich będą przesyłane razem z danymi z czujnika orientacji. Dzięki temu system jest przygotowany na przyszłe rozszerzenia i skalowalność, umożliwiając łatwą integrację dodatkowych funkcjonalności w miarę rozwoju aplikacji.

4.1.3 Przygotowanie danych do wysłania

Dane wszystkich trzech czujników są zapisane w postaci jednego ciągu string, gdzie pierwsze dane dotyczą akcelerometru kolejne żyroskopu a ostatnie danych z czujnika game rotation sensor. Kod odpowiedzialny za konwersję danych z czujników:

```
string converter()
{
    string data;
    data =
        "X: " + acceleration.x + ", Y: " + acceleration.y + ", Z: " +
acceleration.z + ";" +
        " X: " + angularVelocity.x + ", Y: " + angularVelocity.y + ", Z: " +
angularVelocity.z + ";" +
        " X: " + attitudeSensor.x + ", Y: " + attitudeSensor.y + ", Z: " +
attitudeSensor.z + ", W: " + attitudeSensor.w + ";" +
        " " + DateTime.Now.ToString(format);
    return data;

}
```

Analizując kod, można zauważyć, że poza danymi z sensorów została dodana także data, która zawiera dzień i godzinę, jest ona dodana na potrzeby weryfikacji czy wszystkie dane zostają odebrane przez aplikację na telefon.

4.1.4 Przesyłanie danych

Aby przesłać dane z czujników z aplikacji mobilnej do aplikacji komputerowej, konieczne jest utworzenie połączenia UDP. Proces ten obejmuje konfigurację połączenia, przesyłanie danych oraz utrzymywanie stabilnej komunikacji. Poniżej przedstawiono szczegóły implementacji.

- **Inicjalizacja połączenia i przesyłanie danych**

Abu rozpocząć inicjalizację połączenia użytkownik musi wprowadzić dane poprzez wpisanie w pola tekstowe widoczne na rysunku (Rysunek 4) adresu IP, numeru portu serwera oraz częstotliwości z jaką mają być przesyłane dane z aplikacji.

Rysunek 4 Aplikacja na telefon.

Następnie poprzez kliknięcie przycisku "Send Data" połączenie UDP jest inicjowane. Kod odpowiedzialny za utworzenie połączenia wygląda następująco:

- **Rozpoczęcie inicjalizacji połączenia związane z przyciskiem SendData**

```
public void ConnectButton_Clicked()
{
    if (!isSendingData)
    {
        try
        {
            string ipAddress = textAddress.text;
            string portText = textPort.text;

            int port = int.Parse(portText);
            bool isConnected = ConnectToServer(ipAddress, port);
            if (isConnected)
            {
                textStatus.text = "Połączono z serwerem";
                isSendingData = true;
                StartSendingData();
            }
            else
            {
                textStatus.text = "Błąd połączenia z serwerem";
            }
        }
        catch (Exception ex)
        {
            textStatus.text = $"Błąd: Error 001";
        }
    }
}
```

- **Metoda odpowiedzialna za utworzenie połączenia:**

```
public bool ConnectToServer(string ipAddress, int port)
{
    try
    {
        udpClient = new UdpClient(ipAddress, port);
        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
```

- **Wysyłanie danych:**

Po ustanowieniu połączenia, aplikacja rozpoczyna wysyłanie danych z określoną częstotliwością. Proces ten jest realizowany za pomocą metody StartSendingData(), która tworzy nowy wątek odpowiedzialny za wysyłanie danych.

```
private void StartSendingData()
{
    isSendingData = true;
    sendingThread = new Thread(SendDataLoop);
    sendingThread.Start();
}
```

- **Pętla wysyłająca dane:**

Metoda SendDataLoop() jest odpowiedzialna za ciągłe wysyłanie danych do serwera w ustalonych odstępach czasu. Częstotliwość wysyłania danych jest określona przez użytkownika w polu tekstowym textFrequency.

```
private void SendDataLoop()
{
    int freq = int.Parse(textFreqency.text);
    while (isSendingData)
    {
        try
        {
            SendDataToServer(converter());
            Thread.Sleep(freq);
        }
        catch (Exception ex)
        {
            textStatus.text += ex.Message;
        }
    }
}
```

Wyjaśnienie kodu:

- **Metoda ConnectButton_Clicked()**:
  - Pobiera adres IP i port z pól tekstowych.
  - Próbuje nawiązać połączenie z serwerem za pomocą ConnectToServer().
  - W przypadku sukcesu, aktualizuje status i rozpoczyna wysyłanie danych.
- **Metoda ConnectToServer()**:
  - Tworzy nowy obiekt UdpClient z podanym adresem IP i portem.
  - Zwraca true, jeśli połączenie się powiedzie, lub false w przypadku błędu.
- **Metoda StartSendingData()**:
  - Ustawia flagę isSendingData na true i uruchamia nowy wątek SendDataLoop().
- **Metoda SendDataLoop()**:
  - Pobiera częstotliwość wysyłania danych z pola tekstowego textFrequency.
  - W pętli wysyła dane do serwera i czeka określoną ilość czasu (w milisekundach) przed kolejnym wysłaniem danych.

Ten proces zapewnia, że dane z czujników są przesyłane na bieżąco do aplikacji komputerowej, umożliwiając precyzyjne odwzorowanie ruchów. Implementacja ta jest skalowalna i pozwala na łatwe rozszerzenie funkcjonalności w przyszłości.

4.2 Aplikacja na komputer

Aplikacja na komputer będzie odpowiedzialna za odbieranie danych z aplikacji na telefony, deserializowanie ich z powrotem do typu Vector3, Vector3, Quaternion, a następnie przetworzenie tych danych i przypisanie ich do rotacji odpowiednich części ciała awatara w grze. Może ona działać nawet z jednym smartfonem jako czujnikiem, jednakże im więcej smartfonów jest używanych (maksymalnie do czterech), tym lepsze i dokładniejsze jest śledzenie ruchów użytkownika. Aplikacja także została napisana w Unity, co umożliwia łatwą integrację z różnymi komponentami VR oraz zapewnia wysoką wydajność i elastyczność.

Główne zadania aplikacji na komputer obejmują:

1. **Odbiór danych**: Aplikacja nasłuchuje na określonych portach UDP, aby odbierać dane przesyłane przez aplikacje mobilne. Każda kończyna jest przypisana do innego portu, co umożliwia równoczesne odbieranie danych z wielu źródeł.
2. **Deserializacja danych**: Otrzymane dane są deserializowane z formatu, w jakim zostały przesłane, do odpowiednich typów danych (Vector3 dla przyspieszenia i prędkości kątowej, Quaternion dla orientacji).
3. **Przetwarzanie danych i aktualizacja awatara**: Deserializowane dane są przetwarzane, aby odzwierciedlić ruchy użytkownika w wirtualnym środowisku. Aplikacja oblicza nowe pozycje i rotacje dla każdej kończyny awatara na podstawie odebranych danych. Przetworzone dane są używane do aktualizacji rotacji i pozycji odpowiednich części ciała awatara w grze. Dzięki temu ruchy użytkownika są precyzyjnie odwzorowane w wirtualnej rzeczywistości, co zapewnia realistyczne i immersyjne doświadczenie.

4.2.1 Odbiór danych

Odbiór danych w aplikacji komputerowej odbywa się poprzez stworzenie serwera nasłuchującego na określonych portach UDP. Serwer odbiera dane przesyłane przez aplikacje mobilne, a następnie rozsyła je do odpowiednich skryptów na podstawie portu, z którego zostały wysłane.

- **Stworzenie serwera nasłuchującego:**

Serwer nasłuchujący jest uruchamiany na wybranym porcie. Odbiera dane w sposób asynchroniczny, co pozwala na ciągłe nasłuchiwanie bez blokowania głównego wątku aplikacji. Poniższy kod przedstawia implementację serwera nasłuchującego:

```
public UdpServer(int port)
{
    this.port = port;
}

public async Task StartListening()
{
    udpListener = new UdpClient(port);
    Debug.Log($"Server started on port {port}, waiting for connection...");

    try
    {
        while (true)
        {
            UdpReceiveResult result = await udpListener.ReceiveAsync();
            string receivedData = Encoding.UTF8.GetString(result.Buffer);
            DataConnector.Instance.SendData(port, receivedData);
        }
    }
    catch (Exception ex)
    {
        Debug.Log($"Error receiving UDP data on port {port}: {ex.Message}");
    }

}
```

- Rozsyłanie otrzymanych danych:

Po odebraniu danych, są one rozsyłane do odpowiednich skryptów na podstawie portu, z którego zostały wysłane. Każdy port jest przypisany do innej części ciała awatara, co umożliwia proste przypisanie rotacji. Kod odpowiedzialny za rozsyłanie danych wygląda następująco:

```
public void SendData(int port, string data)
{
    switch (port)
    {
```

```
        case 12345:
            {
                DataSentToPort12345(data);
                break;
            };
        case 12346:
            {
                DataSentToPort12346(data);
                break;
            };
        case 12347:
            {
                DataSentToPort12347(data);
                break;
            };
        case 12348:
            {
                DataSentToPort12348(data);
                break;
            };
        default:
            Debug.LogError($"Invalid port number: {port}");
            break;

    }
```

- Subskrypcja danych:

Każdy skrypt odpowiedzialny za poruszanie kończynami nasłuchuje określonego portu i sprawdza, czy na ten port nie otrzymano nowych danych. Subskrypcja danych odbywa się poprzez przypisanie metody obsługującej dane do odpowiedniego portu:

```
private void Subscribe(int port)
{
    switch (port)
    {
        case 12345:
            DataConnector.Instance.DataSentToPort12345 += ProcessData;
            sensorName = "LeftTop ";
            break;
        case 12346:
            DataConnector.Instance.DataSentToPort12346 += ProcessData;
            sensorName = "LeftDown ";
            break;
        case 12347:
            DataConnector.Instance.DataSentToPort12347 += ProcessData;
            sensorName = "RightTop ";
            break;
        case 12348:
            DataConnector.Instance.DataSentToPort12348 += ProcessData;
            sensorName = "RightDown ";
            break;
        default:
            Debug.LogError($"Invalid port number: {port}");
            break;
    }

}
```

Dzięki temu procesowi aplikacja na komputer może odbierać dane wysłane przez wcześniej napisaną aplikację na telefony.

4.2.2 Deserializacja danych

Po uzyskaniu danych z aplikacji mobilnych, dane te są przekazywane do DataConvertera, który konwertuje ciąg znaków na odpowiednie typy: Vector3 dla danych z akcelerometru, Vector3 dla danych z żyroskopu, Quaternion(kwaternion) dla danych z czujnika orientacji (Game Rotation Vector Sensor) oraz DateTime dla znacznika czasu, kiedy dane zostały wysłane. Proces deserializacji odbywa się za pomocą wyrażeń regularnych (regex). Poniższy kod przedstawia, jak dane są przetwarzane przez DataConvertera:

```
Regex regex = new Regex(@"X:\s*(-?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*Y:\s*(-
?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*Z:\s*(-?\d+[\.,]?\d*(?:E[-+]?\d+)?);\s*X:\s*(-
?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*Y:\s*(-?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*Z:\s*(-
?\d+[\.,]?\d*(?:E[-+]?\d+)?);\s*X:\s*(-?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*Y:\s*(-
?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*Z:\s*(-?\d+[\.,]?\d*(?:E[-+]?\d+)?),\s*W:\s*(-
?\d+[\.,]?\d*(?:E[-+]?\d+)?);\s*(\d+:\d+:\d+\.\d+)");

if (match.Success)
 {
     var cultureInfo = new CultureInfo("en-US");
     cultureInfo.NumberFormat.NumberDecimalSeparator = ".";

     accelerometer = new Vector3(
         float.Parse(match.Groups[1].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[2].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[3].Value, NumberStyles.Float, cultureInfo)
     );

     gyroscope = new Vector3(
         float.Parse(match.Groups[4].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[5].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[6].Value, NumberStyles.Float, cultureInfo)
     );

     attitude = new Quaternion(
         float.Parse(match.Groups[7].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[8].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[9].Value, NumberStyles.Float, cultureInfo),
         float.Parse(match.Groups[10].Value, NumberStyles.Float, cultureInfo)
     );

     timestamp = DateTime.Parse(match.Groups[11].Value);

     return (accelerometer, gyroscope, attitude, timestamp);
```

Poprzez wykonanie takiego przetworzenia program posiada dane w postaci takiej, w jakiej zostały one pobrane z czujnika w aplikacji na telefon. Warto zwrócić uwagę, że dane mają również timestamp, który zawiera datę i czas, w którym zostały wysłane. Może to być później wykorzystane w celu weryfikacji, czy wszystkie dane zostały poprawnie odebrane.

4.2.3 Przetwarzanie danych i aktualizacja awatara

W procesie przetwarzania danych skupiamy się na danych attitude, które są danymi z czujnika Game Rotation Vector. Dane te są w postaci Quaternion, co pozwala na precyzyjne odwzorowanie rotacji urządzenia w przestrzeni trójwymiarowej. Dzięki odpowiedniemu dostosowaniu rotacji telefonu możemy uzyskać rotację odpowiadającą aktualnej rotacji odpowiedniej części nogi w rzeczywistości.

- **Wywołanie serializacji:**

```
(_, _, attitude, _) = converter.ParseData(data);
```

- **Metoda dostosowująca rotację:**

```
private Quaternion rightCoordToUnityCord(Quaternion q)
{
    Quaternion originalRotation = new Quaternion(q.x, q.y, q.z, q.w);

    // Quaternion reprezentujący rotację o 180 stopni wokół osi Y
    Quaternion yRotation180 = new Quaternion(0, 0, 1, 0);

    // Mnożenie oryginalnego quaternionu przez rotację wokół osi Y
    return originalRotation * yRotation180;
}
```

- **Dostosowanie rotacji do systemu Unity:**

```
attitude = rightCoordToUnityCord(attitude);
```

- **Inicjalizacja początkowej rotacji:**

```
if (first)
{
    startingPhoneRotation = attitude;
    muscleObject.transform.rotation = startingRotation;
    startingPlayerRotation = muscleObject.transform.rotation;
    first = false;
}
```

- **Obliczenie relatywnej rotacji:**

```
Quaternion relativeRotation = Quaternion.Inverse(startingPhoneRotation) *
attitude;
```

- **Dostosowanie orientacji postaci gracza:**

```
muscleObject.transform.rotation = startingPlayerRotation * relativeRotation;
```

Opis działania kodu:

- **Deserializacja danych**: Otrzymane dane są przetwarzane przez DataConvertera, który konwertuje je na odpowiednie typy, w tym Quaternion dla danych attitude.
- **Dostosowanie rotacji**: Metoda rightCoordToUnityCord przekształca orientację z systemu współrzędnych telefonu na system współrzędnych używany przez Unity. Jest to konieczne, aby zapewnić poprawne odwzorowanie rotacji w środowisku VR.
- **Inicjalizacja początkowej rotacji**: Przy pierwszym odbiorze danych ustawiamy początkową rotację telefonu i postaci gracza. Ta inicjalizacja pozwala na określenie bazowej pozycji, z której będą obliczane dalsze rotacje.
- **Obliczenie relatywnej rotacji**: Relatywna rotacja jest obliczana jako iloczyn odwrotności początkowej rotacji telefonu i bieżącej rotacji. Ta operacja pozwala na uzyskanie względnej zmiany rotacji od momentu rozpoczęcia śledzenia.
- **Dostosowanie orientacji postaci gracza**: Na koniec, obliczona relatywna rotacja jest stosowana do rotacji postaci gracza w grze. Dzięki temu ruchy kończyn dolnych użytkownika są precyzyjnie odwzorowane w środowisku VR.

Dzięki zastosowaniu zmiennej typu bool o nazwie first, możemy w każdej chwili ponownie ustawić początkową rotację telefonu i postaci gracza, co jest później wykorzystywane w przycisku resetowania pozycji.

5 Testowanie i walidacja

W tej sekcji opisane zostaną dwa główne rodzaje testów przeprowadzonych w ramach projektu: testy ilości zgubionych pakietów danych oraz testy wrażeń użytkowników. Pierwszy typ testów ma na celu ocenę niezawodności przesyłu danych za pomocą UDP, natomiast drugi typ testów koncentruje się na subiektywnych odczuciach użytkowników dotyczących dokładności odwzorowania ruchów w środowisku VR.

5.1 Test ilości zgubionych pakietów

Testy ilości zgubionych pakietów mają na celu sprawdzenie, ile pakietów danych zostanie zgubionych podczas przesyłu danych z czujników na aplikację komputerową. Testy zostały wykonane przy użyciu dwóch telefonów: HUAWEI Mate 20 Lite (urządzenie 1) oraz Huawei P20 Lite (urządzenie 2). Testy te mają na celu zrozumienie wpływu różnych modeli telefonów i warunków przesyłu danych na utratę pakietów.

Kod wykorzystany przy testowaniu:

```
void CheckDate(DateTime oldDate, DateTime newDate)
{
   var timeDifference = (newDate - oldDate).TotalMilliseconds;
   if (Math.Abs(timeDifference) > toleranceMs)
   {
      lostDataAmount++;
      lostDataText.text = lostDataAmount.ToString() +" Missing Data";
   }
}
```

Parametry testowe:

- toleranceMs oznacza dozwolone opóźnienie dla danych.

Test Pierwszy

Test przeprowadzono przy wysyłaniu danych z częstotliwością 50 Hz i dozwolonym opóźnieniem 2 sekundy.

Wyniki:

- Telefon pierwszy zgubił 30 pakietów na 5429 rekordów danych.
- Telefon drugi zgubił 8 pakietów na 5390 rekordów danych.

Procent utraconych danych:

Telefon pierwszy: $(\frac{30}{5429})\times 100\% \approx 0.5526\%$

Telefon drugi: $(\frac{8}{5390})\times 100\% \approx 0.1484\%$

Wynik testu przedstawione na (rysunku 5) wskazuje na to, że model telefonu ma znacznie podczas wysyłania danych w dużej częstotliwości.

Rysunek 5 Wynik pierwszego testu na utraty danych 50hz max 2 sec opóźnienia.

- **Test drugi**

Test przedstawione na rysunku (Rysunek 6) przeprowadzono przy tej samej częstotliwości wysyłania danych (50 Hz), ale zwiększono dozwolone opóźnienie pakietu z 2 sekund do 5 sekund.

Rysunek 6 Wynik drugiego testu na utraty danych 50hz max 5 sec opóźnienia.

Wyniki:

- Oba telefony zgubiły 2 pakiety na ponad 5000 poprawnie wysłanych danych.

Procent utraconych danych:

Telefon pierwszy: $(\frac{2}{5041})\times 100\% \approx 0.0397\%$

Telefon drugi: $(\frac{2}{5041})\times 100\% \approx 0.0389\%$

Wnioski: Różnica między wynikami testu pierwszego i drugiego pokazuje, że wiele pakietów uznawanych w teście pierwszym za stracone, faktycznie dociera, ale z opóźnieniem. Prawdziwa ilość zgubionych pakietów jest bardzo niska, około 0.04%.

- **Test trzeci**

Test przeprowadzono przy wysyłaniu danych z częstotliwością 100 Hz i dozwolonym opóźnieniem 2 sekundy.

Wyniki:

- Telefon pierwszy zgubił 12 pakietów na 6029 rekordów danych.
- Telefon drugi zgubił 14 pakietów na 6024 rekordów danych.

Procent utraconych danych:

Telefon pierwszy: $(\frac{12}{6029})\times 100\% \approx 0.1992\%$

Telefon drugi: $(\frac{14}{6024})\times 100\% \approx 0.2325\%$

Wyniki testu trzeciego (Rysunek 7) pokazują, że zwiększenie częstotliwości wysyłania danych prowadzi do większej utraty pakietów. Jednak różnica między testami pierwszym i trzecim może również wskazywać na wpływ różnych warunków sieciowych i specyfikacji telefonów.

Rysunek 7 Wynik trzeciego testu na utraty danych 100hz max 2 sec opóźnienia.

- **Test czwarty**

Test przeprowadzono przy wysyłaniu danych z częstotliwością 100 Hz i dozwolonym opóźnieniem 5 sekund.

Wyniki:

- Telefon pierwszy zgubił 5 pakietów na 6355 rekordów danych.
- Telefon drugi zgubił 4 pakiety na 6415 rekordów danych.

Procent utraconych danych:

Telefon pierwszy: $(\frac{5}{6355})\times 100\% \approx 0.0786\%$

Telefon drugi: $(\frac{4}{6415})\times 100\% \approx 0.0623\%$

Pomimo że test trzeci wykazał znaczną ilość pakietów utraconych przez telefon pierwszy (Rysunek 8), poprzez zwiększenie dozwolonego opóźnienia o 3 sekundy ilość strat została w znacznym stopniu zredukowana. Wyniki wskazują, że większość pakietów uznawanych za stracone w rzeczywistości dociera z opóźnieniem. Telefon drugi wykazał mniejszą ilość straconych danych, ale oba wyniki są gorsze niż przy teście na częstotliwości 50 Hz. Wraz ze zwiększeniem częstotliwości przesyłu, dane zaczynają nie docierać do celu.

Rysunek 8 Wynik czwartego testu na utraty danych 100hz max 5 sec opóźnienia.

5.2 Test dokładności odwzorowania ruchów

W tej sekcji zostaną opisane testy dokładności odwzorowania ruchów, które miały na celu sprawdzenie precyzji mapowania rotacji czujników do awatara w środowisku VR. Testy te koncentrowały się na ocenie, jak dokładnie system odwzorowuje ruchy użytkownika na awatarze, oraz na identyfikacji potencjalnych problemów związanych z precyzją i stabilnością odwzorowania.

Zakres rotacji czujników:

Czujniki zastosowane w systemie mają zakres rotacji wynoszący 360 stopni. Jednakże, podczas normalnego użytkowania, gdy czujniki są przypięte do nóg, nie przewiduje się korzystania z pełnego zakresu rotacji. Zakresy rotacji czujników zostały dokładnie sprawdzone, aby upewnić się, że w typowych scenariuszach użytkowania (Wychylenie nogi do przodu tyłu, wychylenie nogi na bok, kopnięcie nogą oraz przysiadu) system odwzorowuje ruchy w sposób precyzyjny. Na rysunku (Rysunek 9) przedstawiono jak wygląda testowane wychylenie nogi na bok, w tym przypadku prawej.

Rysunek 9 Zdjęcie z testów wysunięcia nogi na bok.

Testy te obejmowały kilkukrotne powtórzenia następujących ruchów:

- Wychylenie nogi do przodu tyłu.
- Wychylenie nogi na bok.
- Kopnięcie nogą.
- Przysiad.

Na podstawie tych testów można stwierdzić, że aplikacja precyzyjnie odwzorowuje ruchy, nawet przy wielokrotnych powtórzeniach. Obserwacje wykazały, że:

- Minimalne odchylenia rotacji: Czasami po wykonaniu ruchu rotacja może być minimalnie większa lub mniejsza niż powinna, ale jest to automatycznie korygowane w kolejnych przesyłach danych z czujników. Powoduje to, że noga, mimo braku ruchu użytkownika, może lekko korygować swoją rotację.
- Luźne umocowanie telefonów: Niektóre błędy w dokładności były spowodowane luźnym umocowaniem telefonów na nodze, co powodowało, że telefon obracał się w nieoczekiwany sposób. Poprawne zamocowanie czujników jest kluczowe dla utrzymania wysokiej precyzji odwzorowania ruchów.

Wnioski z testów:

Podczas tych testów zaobserwowano, że dane z czujników płynnie mapują ruchy, nawet w momentach, w których kamera nie nadążała za ruchem (Rysunek 10). Jest to dowód na wysoką precyzję systemu, mimo drobnych problemów z dokładnością wynikających głównie z mechanicznych aspektów mocowania czujników.

Rysunek 10 Zdjęcie z testów zakresu ruchów.

Podsumowując, testy dokładności odwzorowania ruchów potwierdziły, że aplikacja jest w stanie precyzyjnie odwzorować ruchy kończyn dolnych użytkownika, co jest kluczowe dla zapewnienia realistycznych i immersyjnych doświadczeń VR. Mimo pewnych drobnych odchyleń i konieczności dokładnego mocowania czujników, system działa zgodnie z oczekiwaniami i spełnia wymogi funkcjonalne projektu.

5.3 Testy praktyczne użytkownika

Test ten polega na ocenie dokładności odwzorowania ruchów przez użytkownika. Test odbywa się poprzez włączenie programu i subiektywną ocenę użytkownika, czy ruchy są dokładnie odwzorowane w środowisku VR. Użytkownicy sprawdzają, czy ich ruchy kończyn dolnych są płynne i realistycznie odwzorowane na awatarze w VR. Test ten pomaga ocenić subiektywne wrażenia z użytkowania systemu i jego ogólną użyteczność.

Aby dokładnie przetestować odwzorowanie ruchów, została stworzona osobna scena w Unity (Rysunek 11), która symuluje boisko piłki nożnej. W tej scenie gracz może poprzez kliknięcie przycisku na interfejsie graficznym włączyć mini grę. W tej grze piłki nożne pojawiają się w losowych miejscach na boisku, a gracz stoi na bramce. Zadaniem gracza jest nie dopuścić, aby piłka wleciała do bramki, wykorzystując mapowanie nóg przez aplikację.

Rysunek 11 Zdjęcie sceny w aplikacji na telefon do testowania projektu.

Kroki testowania:

1. Użytkownik zakłada smartfony z aplikacją na kończynach dolnych.
2. Użytkownik uruchamia aplikację komputerową i wchodzi na scenę symulującą boisko piłki nożnej.
3. Użytkownik klika przycisk na interfejsie graficznym, aby uruchomić mini grę.
4. Piłki nożne pojawiają się losowo na boisku, a użytkownik musi poruszając nogami, nie dopuścić do wpadnięcia piłki do bramki.
5. Użytkownik ocenia, czy ruchy jego kończyn dolnych są dokładnie odwzorowane w środowisku VR na awatarze.

Wnioski: Test został przeprowadzony na osobach niezaznajomionych z technologią VR, lecz każdy z nich potwierdził, że mapowanie i ruch odbywają się w sposób realistyczny. Użytkownicy stwierdzili, że ruchy ich kończyn są dokładnie odwzorowane w VR, co znacznie poprawia immersję i jakość doświadczeń VR. Jednakże zauważono, że czasami kończyny drgają, co może być spowodowane luźnym przytwierdzeniem telefonów do nóg lub zbyt niską częstotliwością przesyłu danych.

5.4 Podsumowanie wyników testów

Testy ilości zgubionych pakietów wykazały, że przy zwiększeniu częstotliwości przesyłania danych rośnie liczba utraconych pakietów, jednak zwiększenie dozwolonego

opóźnienia znacząco redukuje te straty, wskazując na docieranie wielu pakietów z opóźnieniem. Równocześnie, test odwzorowania ruchów przeprowadzony na osobach niezaznajomionych z technologią VR potwierdził, że mapowanie i ruch są realistycznie odwzorowane w środowisku VR. Uczestnicy zgodnie stwierdzili, że aplikacja precyzyjnie śledzi ich ruchy, co dowodzi, że spełnia ona wymogi funkcjonalne projektu.

6 Podsumowanie i wnioski

Projekt wykazał, że przy wykorzystaniu czujników rotacyjnych w telefonach można precyzyjnie odwzorować ruch kończyn dolnych. Podejście przedstawione w pracy z powodzeniem osiągnęło zamierzone cele, dostarczając funkcjonalne i precyzyjne rozwiązanie do odwzorowania ruchów kończyn dolnych w VR. Wykryte ograniczenia i trudności wskazują na obszary, które można jeszcze usprawnić, co otwiera szerokie możliwości dla dalszego rozwoju i optymalizacji systemu.

6.1 Osiągnięcia projektu

Projekt osiągnął swoje główne cele, które obejmowały stworzenie systemu precyzyjnego odwzorowania ruchu kończyn dolnych w środowisku wirtualnej rzeczywistości (VR). Użycie czujników Game Rotation Vector zamontowanych na smartfonach oraz komunikacji UDP pozwoliło na uzyskanie dokładnych i płynnych ruchów nóg użytkownika. Aplikacje mobilne i komputerowe zostały skutecznie zintegrowane, umożliwiając realistyczne doświadczenia VR. Wdrożenie systemu zostało potwierdzone przez testy, które wykazały niezawodność i zgodność z wymogami funkcjonalnymi projektu. Ponadto, projekt umożliwia działanie z różną liczbą smartfonów, od jednego do czterech, co pozwala na elastyczne dostosowanie poziomu precyzji śledzenia ruchów w zależności od dostępnych zasobów.

6.2 Ograniczenia i trudności

Podczas realizacji projektu napotkano kilka ograniczeń i trudności. Głównym wyzwaniem było zapewnienie niskiego opóźnienia i minimalnej utraty pakietów danych przy przesyłaniu za pomocą UDP. Chociaż zwiększenie dozwolonego opóźnienia pomogło zmniejszyć straty, nadal występowały pewne opóźnienia, które mogły wpływać na płynność odwzorowania ruchów. Konieczność stosowania wielu smartfonów jako czujników może być niewygodna dla użytkowników i wymaga dalszej optymalizacji. Ponadto, złożoność kalibracji i ustawień początkowych mogła stanowić wyzwanie dla użytkowników mniej zaznajomionych z technologią VR. Problemy związane z drganiami kończyn, choć sporadyczne, wskazują na potrzebę dalszych badań i optymalizacji algorytmów przetwarzania danych z czujników.

6.3 Możliwości rozwoju projektu

Projekt oferuje wiele możliwości rozwoju i dalszych badań. Jednym z kierunków rozwoju może być integracja bardziej zaawansowanych czujników ruchu lub opracowanie dedykowanych urządzeń, które mogłyby zastąpić smartfony, oferując lepszą precyzję i wygodę użytkowania. Możliwe jest również dalsze doskonalenie algorytmów przetwarzania danych, aby zredukować opóźnienia i poprawić płynność odwzorowania ruchów. Rozszerzenie funkcjonalności systemu o dodatkowe czujniki na innych częściach ciała mogłoby umożliwić pełne śledzenie ruchów całego ciała w VR, co znacznie zwiększyłoby realizm i immersję doświadczeń VR. Dodatkowo, badania nad optymalizacją komunikacji sieciowej mogłyby poprawić niezawodność przesyłu danych i zmniejszyć liczbę zgubionych pakietów, co jest kluczowe dla aplikacji czasu rzeczywistego.

7 Bibliografia

1. Virtual Reality (VR). *interaction-design.* [Online] 10 06 2024. https://www.interaction-design.org/literature/topics/virtual-reality.

2. Pimax What is VR? Virtual Reality Explained. *pimax.* [Online] 10 06 2024. https://pimax.com/blogs/blogs/what-is-vr-virtual-reality-explained.

3. Lowood Henry E. virtual reality. *britannica.* [Online] 10 06 2024. https://www.britannica.com/technology/virtual-reality.

4. Studiobinder what is virtual reality. *studiobinder.* [Online] 10 06 2024. https://www.studiobinder.com/blog/what-is-virtual-reality/.

5. Wikipedia User Datagram Protocol . *wikipedia.* [Online] 10 06 2024. https://en.wikipedia.org/wiki/User_Datagram_Protocol.

6. Principles of Motion Sensing. *wikipedia.* [Online] 10 06 2024. https://en.wikipedia.org/wiki/Principles_of_Motion_Sensing.

7. Virtual reality motion tracking technology. *servreality.* [Online] 10 06 2024. https://servreality.com/blog/virtual-reality-motion-tracking-technology/.

8. What sensors are used in ar vr systems faq. *sensortips.* [Online] 10 06 2024. https://www.sensortips.com/featured/what-sensors-are-used-in-ar-vr-systems-faq/.

9. Developer.android sesnsor_motion. [Online] [Zacytowano: 10 06 2024.] https://developer.android.com/develop/sensors-and-location/sensors/sensors_motion.

10. Developer.android sensors_position. *developer.android.* [Online] 10 06 2024. https://developer.android.com/develop/sensors-and-location/sensors/sensors_position?hl=pl..

11. User datagram protocol udp. *cloudflare.* [Online] 10 06 2024. https://www.cloudflare.com/learning/ddos/glossary/user-datagram-protocol-udp/.

12. Eater Grant Sanderson Technology by Ben. eater quaternions. *eater.* [Online] 10 06 2024. https://eater.net/quaternions.

13. Mysticle The. *youtube.* [Online] 10 06 2024. https://www.youtube.com/watch?v=6JqEJ17ul9k .

14. Dog Frankly. Driver4VR: Full Body Tracking. *play.google.* [Online] 10 06 2024. https://play.google.com/store/apps/details?id=com.franklydog.driver4vr&pli=1.

15. Vive tracker3. *vive.* [Online] 10 06 2024. https://www.vive.com/uk/accessory/tracker3/.

16. Developer.oculus move body tracking. *developer.oculus.* [Online] 10 06 2024. https://developer.oculus.com/documentation/unity/move-body-tracking/?locale=pl_PL.

17. Calliepepper. Slimevr101. *docs.slimevr.* [Online] 10 06 2024. https://docs.slimevr.dev/slimevr101.html.

18. SlimeVR. Github SlimeVR Server. *github.* [Online] 10 06 2024. https://github.com/SlimeVR/SlimeVR-Server/tree/main.

19. Github April Tag VR FullBody Tracker. *github.* [Online] 10 06 2024. https://github.com/ju1ce/April-Tag-VR-FullBody-Tracker/wiki.

8 Spis rysunków