LABORATORIUM 1
Środowisko XR i audyt urządzeń

Unity • OpenXR • XR Interaction Toolkit • Git • Quest • VITURE • Android

Problem inżynierski

Jak zbudować powtarzalne środowisko XR, w którym ta sama scena zachowuje poprawną skalę i orientację niezależnie od sposobu uruchomienia, a możliwości sprzętu są ustalane na podstawie obserwacji i API, nie nazwy handlowej?

Parametry laboratorium

Parametr	Wartość
Tryb pracy	pary; stanowiska sprzętowe rotacyjne
Rezultat	uruchomiona scena bazowa, pomiar bazowy i karta audytu urządzeń
Oddanie	commit + evidence/lab01.md; bez tradycyjnego sprawozdania
Wersja instrukcji	1.1 • 21 lipca 2026 r.

System oceniania

Ocena wynika z osiągnięcia kolejnych checkpointów . Każdy checkpoint weryfikuje prowadzący w ciągu maksymalnie 60 sekund. Zakres prac do osiągnięcia oceny 3.0 z laboratorium 1, jest wymagany do podejścia do laboratorium 2.

Ocena	Główne wymagania
3.0	Scena działa, geometria 1m, pomiar FPS, diagnoza - zaliczenie Lab
3.5	Transformacje lokalne/globalne, analiza skalowania CAD
4.0	Pełny audyt: Quest, VITURE, Android (DoF, wejście, API)
4.5	Porównanie runtime'ów, interpretacja ograniczeń, rekomendacje
5.0	Optymalizacja, eksploracja granic, scenariusz praktyczny, wnioski

Walidacja: Numer indeksu i wariant diagnostyczny

Aby uniknąć kopiowania wyników między parami, każdy student wpisuje swój numer indeksu do skryptu diagnostycznego. Wariant błędu w Części D wynika ze wzoru:

$$\text{wariantDiagnostyczny} = (\text{numer_indeksu_1} + \text{numer_indeksu_2}) \bmod 4$$

Wynik: wariant A, B, C lub D (każda para ma inny błąd do naprawienia)

Evidence/lab01.md musi zawierać numery indeksów obu studentów. Prowadzący sprawdza, czy przydzielony wariant zgadza się z wyliczonym ze wzoru.

Wymagania wstępne

Przed zajęciami

- Unity Hub oraz wersja Unity wskazana w pliku ProjectSettings/ProjectVersion.txt
- Git w wierszu poleceń lub zatwierdzony klient graficzny
- Minimum 15 GB wolnego miejsca na dysku
- Konto w systemie udostępniającym repozytorium kursu
- Podstawy C#: komponent MonoBehaviour, metoda Update(), pole publiczne/serializowane
- Podstawy geometrii 3D: wektor, obrót, układ lokalny i globalny

Pakiety wymagane w projekcie

Pakiet	Identyfikator	Rola
XR Plug-in Management	com.unity.xr.management	inicjalizacja i wybór dostawcy XR

OpenXR Plugin	com.unity.xr.openxr	połączenie aplikacji z aktywnym runtime OpenXR
XR Interaction Toolkit	com.unity.xr.interaction.toolkit	XR Origin, interakcje i symulator
Input System	com.unity.inputsystem	akcje wejścia dla symulatora i urządzeń

Numery wersji pobiera się z Packages/manifest.json i Packages/packages-lock.json. Instrukcja celowo nie narzuca numeru „latest": źródłem prawdy jest zamrożone repozytorium kursu.

Bezpieczeństwo i higiena pracy

- Przed założeniem gogli sprawdź wolną przestrzeń i aktywną granicę bezpieczeństwa; nie wykonuj gwałtownych kroków poza wyznaczony obszar.
- Przewody zasilające i sygnałowe prowadź poza strefą ruchu. Jedna osoba korzysta ze sprzętu, druga obserwuje otoczenie.
- Przerwij ekspozycję po wystąpieniu nudności, zawrotów głowy, bólu oczu, dezorientacji lub utraty równowagi. Objawów nie traktuje się jako elementu zadania.
- Czyść powierzchnie mające kontakt ze skórą zgodnie z procedurą laboratorium; nie stosuj płynów bezpośrednio na soczewki ani elektronikę.
- Nie uruchamiaj passthrough ani nagrywania obrazu bez poinformowania osób znajdujących się w polu widzenia urządzenia.
- Nie loguj się prywatnym kontem na urządzeniu laboratoryjnym i nie kopiuj danych biometrycznych do repozytorium.

Podstawy teoretyczne

4.1. Warstwy uruchomieniowe XR

OpenXR jest otwartym standardem API, który oddziela aplikację od konkretnego producenta urządzenia. Aplikacja komunikuje się z jednym aktywnym runtime OpenXR; rozszerzenia są opcjonalne i mogą być dostępne tylko na części urządzeń lub platform [1], [2]. W Unity pakiet XR Plug-in Management zarządza cyklem życia dostawcy XR, a XR Interaction Toolkit dostarcza komponenty interakcji wysokiego poziomu.

Warstwa	Rola
Źródło pozy i wejścia	Gogle, kontrolery, klawiatura i mysz
Runtime OpenXR	Pośrednik między aplikacją i urządzeniem
OpenXR Plugin	Interfejs w Unity do runtime OpenXR
XR Interaction Toolkit	Komponenty wysokopoziomowe (XR Origin, interakcje)
Warstwa aplikacji	Scena, logika gry, UI

Zasada diagnostyczna: Brak ruchu kamery nie oznacza automatycznie błędu XR Origin. Należy kolejno sprawdzić: źródło wejścia → aktywny runtime loader OpenXR → aktywne akcje Input System → powiązania komponentów w scenie.

4.2. 3DoF i 6DoF

Model	Składowe pozy	Co można zaobserwować
3DoF	Obrót: roll, pitch, yaw	Zmiana kierunku patrzenia bez translacji głowy
6DoF	Obrót + położenie x, y, z	Obrót oraz przesunięcie głowy/kontrolera w przestrzeni

Klasyfikacja dotyczy konkretnego kanału danych dostępnego aplikacji. To samo urządzenie może wyświetlać obraz, ale nie udostępniać aplikacji pozy głowy albo udostępniać ją wyłącznie przez określony tryb, oprogramowanie lub SDK.

4.3. Układy współrzędnych i skala

Unity stosuje oś X w prawo, Y w górę i Z do przodu; jednostka świata powinna odpowiadać 1 m. Pozycja lokalna obiektu jest wyrażona względem rodzica, a pozycja globalna względem początku świata sceny. Transformacja hierarchiczna ma postać:

$$p_{world} = T_{parent} \cdot p_{local}$$

W praktyce T obejmuje translację, obrót i skalę. Niejednostkowa skala rodzica może zmieniać rozmiar i kierunek przesunięcia potomka, dlatego w hierarchiach XR skala XR Origin powinna zwykle pozostać równa (1, 1, 1).

4.4. Pomiar czasu klatki

Częstotliwość klatek jest odwrotnością średniego czasu klatki: f = 1/t. Dla 72 Hz budżet wynosi około 13,89 ms, dla 90 Hz - 11,11 ms, a dla 120 Hz - 8,33 ms. Wynik z trybu Play w Edytorze obejmuje narzut narzędzi i nie jest równoważny pomiarowi samodzielnej kompilacji na urządzeniu. W raporcie należy zatem podać środowisko pomiaru.

Procedury: Checkpointy i wymagania

Plan pracy

Etap	Dowód ukończenia
Pretest i model przepływu danych	Odpowiedzi + szkic warstw
Repozytorium, wersje, uruchomienie symulatora	Scena działa bez błędów krytycznych
Skala, osie, transformacje i metryka czasu klatki	Obiekt 1m + wynik pomiaru FPS
Rotacyjny audyt Quest, VITURE i Android	Uzupełniona karta audytu
Diagnoza celowego błędu	Przyczyna, poprawka, commit
Oddanie i exit ticket	Evidence/lab01.md + odpowiedź

Checkpoint 3.0

Wymagania do zaliczenia:

✓ Scena uruchamia się w Play Mode bez błędów krytycznych

✓ XR Origin z symulatorem - można sterować głową w 6DoF (obrót i translacja)

✓ Cube_1m ma rzeczywisty wymiar 1 m × 1 m × 1 m (weryfikacja w Bounds lub Collider)

✓ Osie XYZ są czytelne i mają długość 1 m

✓ Zarejestowany czas klatki: trzy odczyty (10s, 20s, 30s) i obliczona mediana

✓ Pomiar zawiera kontekst: środowisko (Edytor/build), platforma, rozdzielczość Game View, GPU, runtime

✓ Diagnoza: objaw mierzalnie, dwie hipotezy, test rozstrzygający, poprawka, wynik po

✓ Evidence/lab01.md zawiera: numery indeksów, wersje, rozmiar geometrii, medianę FPS, wariant diagnostyczny

Część A. Uruchomienie projektu

A.1. Pretest (10 minut, indywidualnie)

Pretest ma charakter diagnostyczny i nie jest punktowany. Po oddaniu testu porównaj uzasadnienie odpowiedzi z konfiguracją własnej sceny.

1. Uporządkuj warstwy od sprzętu do obiektu sceny: XRI, runtime OpenXR, urządzenie, OpenXR Plugin, XR Origin.
2. Wyjaśnij jednym zdaniem różnicę między 3DoF i 6DoF.
3. Obiekt ma skalę (1, 1, 1), a jego rodzic (0.01, 0.01, 0.01). Jaki będzie przybliżony rozmiar świata obiektu o boku 1 jednostki?
4. Podaj dwie przyczyny, dla których FPS zmierzony w Edytorze nie powinien być bezpośrednio porównywany z FPS kompilacji na goglach.

A.2. Pobranie i identyfikacja wersji (10 minut)

1. Przejdź do katalogu roboczego. Nie umieszczaj projektu w katalogu synchronizowanym w chmurze ani na dysku sieciowym.
2. Sklonuj repozytorium kursu: git clone <ADRES_REPOZYTORIUM> xr-workcell cd xr-workcell git fetch --all --tags git switch lab01-start
3. Utwórz gałąź pary. Stosuj małe litery, bez spacji i polskich znaków: git switch -c team-<nr>/lab01-<nazwisko1>-<nazwisko2>
4. Odczytaj wersję Unity z ProjectSettings/ProjectVersion.txt. Zainstaluj tę wersję w Unity Hub; nie akceptuj automatycznej migracji do innej wersji.
5. Otwórz projekt z Unity Hub. Pierwszy import może trwać kilka minut. Po zakończeniu nie powinny występować czerwone komunikaty kompilatora w Console.

Wariant awaryjny bez repozytorium: Utwórz projekt 3D URP w wersji wskazanej przez prowadzącego. W Package Manager zainstaluj XR Plug-in Management, OpenXR Plugin, XR Interaction Toolkit i Input System. Z pakietu XRI zaimportuj próbki Starter Assets oraz XR Device Simulator/XR Interaction Simulator. Zapisz scenę jako Assets/Scenes/Lab01_Baseline.unity.

A.3. Kontrola konfiguracji XR (5 minut)

Window → Package Manager: sprawdź obecność czterech wymaganych pakietów.

Edit → Project Settings → XR Plug-in Management: sprawdź, czy dla bieżącej platformy wybrano OpenXR.

Edit → Project Settings → OpenXR: otwórz Project Validation. Zapisz ostrzeżenia; popraw tylko te, które wskazuje instrukcja lub prowadzący.

Edit → Project Settings → Player → Active Input Handling: użyj konfiguracji zapisanej w projekcie. Po zmianie Input System uruchom ponownie Edytor, jeśli pojawi się monit.

A.4. XR Origin i symulator (10 minut)

1. Otwórz scenę bazową. Usuń dodatkową Main Camera tylko wtedy, gdy prefab XR Origin zawiera już kamerę oznaczoną tagiem MainCamera.
2. Dodaj przygotowany prefab XRI. Dla XRI 3.x użyj prefabu XR Interaction Setup ze Starter Assets albo równoważnej konfiguracji w repozytorium.
3. Dodaj prefab symulatora. Użyj XR Device Simulator lub XR Interaction Simulator zgodnie z wersją XRI. Nie twórz własnych mapowań klawiatury, jeżeli próbka dostarcza gotowe akcje.
4. Uruchom Play Mode. Porusz wirtualną głową i sprawdź translację oraz obrót. Jeśli symulator przechwytuje kursor, użyj skrótu wyświetlonego w jego panelu pomocy.

PUNKT KONTROLNY A: W Hierarchy istnieje dokładnie jedna aktywna kamera główna. XR Origin i symulator są aktywne. W Play Mode można niezależnie zmienić orientację oraz położenie głowy, a Console nie zawiera błędów kompilacji.

Część B. Scena wzorcowa i pomiar bazowy

B.1. Wzorzec metryczny (10 minut)

1. Utwórz pusty obiekt WorkcellOrigin. Ustaw Position = (0, 0, 0), Rotation = (0, 0, 0), Scale = (1, 1, 1).
2. Dodaj podłoże. Utwórz Plane lub siatkę 4 m × 4 m. Poziom podłoża ustaw na Y = 0.
3. Dodaj Cube_1m. Ustaw Scale = (1, 1, 1) i Position = (0, 0.5, 2). Dolna ściana sześcianu powinna stykać się z podłożem.
4. Dodaj reprezentację osi. X - czerwony, Y - zielony, Z - niebieski; każda oś ma długość 1 m. Oznacz osie obiektami TextMeshPro albo etykietami w widoku Scene.
5. Sprawdź skalę. Użyj narzędzia pomiarowego Edytora lub porównaj wartości Bounds/Collider. Zapisz zrzut z widocznym Inspector.

B.2. Transformacja lokalna i globalna (15 minut)

Przed uruchomieniem przewidź wynik na papierze, a dopiero potem sprawdź go w Inspectorze:

1. Utwórz obiekt Parent. Position = (2, 0, 1), Rotation Y = 90°, Scale = (1, 1, 1).
2. Utwórz potomka Child. Local Position = (0, 0, 1).
3. Zapisz przewidywaną pozycję globalną Child. Uwzględnij obrót Parent!
4. Następnie odczytaj Transform.position w Play Mode lub przez skrypt diagnostyczny.
5. Zmień skalę Parent na (0.01, 0.01, 0.01). Wyjaśnij, dlaczego taka pomyłka jest szczególnie niebezpieczna po imporcie modeli CAD.

B.3. Rejestrator czasu klatki (15 minut)

Dodaj skrypt Assets/Scripts/BaselineFrameTime.cs do pustego obiektu Diagnostics:

```
using UnityEngine;
public class BaselineFrameTime : MonoBehaviour{
  [Range(0.01f, 1f)]
  public float smoothing = 0.1f;
  private float smoothedDelta;
  int pairIndexSum = 0; // WPISZ TUTAJ: suma numerów indeksów pary (np. 123456 + 654321 = 777777)
  void Start()
```

```
    {
        smoothedDelta = Time.unscaledDeltaTime;
    }
    void Update()
    {
        smoothedDelta = Mathf.Lerp(smoothedDelta,Time.unscaledDeltaTime, smoothing);
    }
    void OnGUI()
    {
        float ms = smoothedDelta * 1000f;
        float fps = smoothedDelta > 0f ? 1f / smoothedDelta : 0f;
        GUI.Label(new Rect(12, 12, 320, 28),
        $"{fps:F1} FPS | {ms:F2} ms | seed={pairIndexSum}");
    }
}
```

1. Wpisz do zmiennej pairIndexSum sumę numerów indeksów pary
2. Uruchom Play Mode przez 30 sekund. Nie zmieniaj rozmiaru Game View podczas próby.
3. Zapisz trzy odczyty FPS i ms po około 10, 20 i 30 sekund
4. Oblicz medianę każdej miary
5. Zapisz kontekst pomiaru: Edytor/build, platforma, rozdzielczość Game View, aktywny symulator/runtime i urządzenie GPU

PUNKT KONTROLNY B: Cube_1m ma rzeczywisty wymiar 1 m, osie są jednoznaczne, wynik transformacji Child zgadza się z przewidywaniem po uwzględnieniu obrotu rodzica, a evidence/lab01.md zawiera medianę FPS i czasu klatki wraz z kontekstem pomiaru.

Checkpoint 3.5

Wymagania dodatkowe (oprócz checkpointu 3.0):

1. Transformacja lokalna vs globalna: Child umieszczony względem Parent
2. Przewidywana pozycja globalna wyliczona ręcznie i weryfikacja poprzez Transform.position (uwzględniając obrót Parent)
3. Zmiana skali Parent na (0.01, 0.01, 0.01) - obserwacja efektu na Child
4. Analiza ryzyka: wyjaśnienie, dlaczego skala rodzica jest szczególnie niebezpieczna po imporcie modeli CAD
5. Wniosek: procedury zabezpieczające przy imporcie geometrii

Weryfikacja (prowadzący, max 60s):
✓ Checkpoint 3.0: wszystkie kryteria (4/4)
✓ Transformacja Child: obliczenie ręczne zgadza się z pomiarem
✓ Analiza skalowania CAD: wyjaśnienie ryzyka
Zaliczenie: 2/3 dodatkowe kryteria (+ checkpoint 3.0)

Checkpoint 4.0

Dodatkowe wymagania (oprócz checkpointu 3.0):

✓ Rotacyjny audyt trzech urządzeń (Quest 3, VITURE, telefon Android)

✓ Dla każdego urządzenia: identyfikacja DoF pozy, dostępne źródła wejścia, aktywny runtime/API

✓ Jedno istotne ograniczenie dla każdego urządzenia

✓ Rozróżnienie: „potwierdzone" (obserwacja/API) vs „niedostępne" (funkcja nie istnieje) vs „niezweryfikowano" (brak dostępu do testu)

✓ Źródło dowodu dla każdego wpisu

Część C. Rotacyjny audyt urządzeń (40 minut)

Każda para odwiedza trzy stanowiska. Jeżeli czas lub liczba zespołów nie pozwala na pełną rotację, prowadzący prezentuje urządzenie na ekranie, a zespoły uzupełniają kartę na podstawie tej samej obserwacji. Nie przypisuj funkcji na podstawie nazwy produktu - oznacz „niezweryfikowano", gdy brak bezpośredniego dowodu.

C.1. Stanowisko Quest 3

1. Sprawdź translację i obrót głowy: wykonaj małe przesunięcie w bok, w przód oraz obrót. Zaobserwuj sposób, w jaki kamera się porusza.
2. Sprawdź dostępne źródła wejścia: kontrolery, śledzenie dłoni lub inne; zapisz tylko te faktycznie widoczne dla aplikacji (zweryfikuj poprzez Input System czy API).
3. Sprawdź, czy scena działa immersyjnie, czy wykorzystuje passthrough. Nie wnioskuj z samego obrazu, czy aplikacja otrzymuje obraz kamer - sprawdź w kodzie lub ustawieniach.
4. Odczytaj nazwę aktywnego runtime/profilu i tryb uruchomienia: Link, build urządzenia, czy symulator.
5. Zapisz jedno ograniczenie: np. dostępność rozszerzenia, konieczność konfiguracji, opóźnienie lub brak funkcji w użytym API.

C.2. Stanowisko VITURE

1. Ustal, czy urządzenie jest widziane jako zewnętrzny ekran, czy aplikacja otrzymuje również dane pozy/orientacji.
2. Porównaj obraz head-locked (przesuwany z głową) z trybem przestrzennym (statycznym w świecie), jeżeli jest dostępny w przygotowanym oprogramowaniu.
3. Sprawdź, jakie dodatkowe oprogramowanie, akcesorium lub SDK jest wymagane dla obserwowanej funkcji.
4. Zapisz ograniczenie istotne dla aplikacji przemysłowej: wejście, pole ruchu, stabilność, ergonomia lub zależność od hosta.

C.3. Stanowisko Android

1. Zidentyfikuj model telefonu i wersję systemu. Nie zapisuj numeru seryjnego ani identyfikatorów użytkownika.
2. Sprawdź dostępność usług ARCore lub uruchomionego testu kompatybilności (aplikacja ARCore dostarczona przez Google).
3. Zaobserwuj, czy aplikacja estymuje ruch 6DoF oraz czy wykrywa płaszczyznę; nie jest to jeszcze laboratorium implementacji AR, jedynie sprawdzenie możliwości.
4. Zapisz dostępność funkcji dodatkowych jako: potwierdzona / niedostępna / niezweryfikowana. Dotyczy to zwłaszcza głębi (depth sensing) i okluzji.

C.4. Karta audytu

Urządzenie	DoF pozy głowy	Wejście	API / Runtime	Ograniczenie / Status
Quest 3	3DoF/6DoF?	Kontrolery/dłonie?	OpenXR / Meta	Rozszerzenie / Latency / ...
VITURE	3DoF/6DoF?	Mysz/Joystick?	SDK X / USB	Zależność od hosta / ...
Android	3DoF/6DoF?	Touch / Gamepad?	ARCore v.X	Depth / Occlusion / ...
Symulator XRI	Symulowane: 6DoF	Klawiatura/mysz: WASD/Look	XRI v.X	Brak wejścia haptycznego / ...

PUNKT KONTROLNY C: Dla każdego urządzenia wskazano źródło dowodu: obserwacja, ekran ustawień, log lub wynik API. Brak funkcji i brak możliwości jej zweryfikowania są zapisane jako dwa różne stany.

Część D. Zadanie diagnostyczne

Wyznaczenie wariantu:

$$\text{wariant} = (\text{numer_indeksu_1} + \text{numer_indeksu_2}) \bmod 4$$

Wariant	Objaw	Przykładowe hipotezy do rozróżnienia
A	Obiekt wzorcowy jest 100 razy za mały	Skala rodzica; import cm→m; skala prefabrykatu
B	Ruch do przodu daje ruch w bok lub przeciwnie	Zamiana osi; obrót rodzica; błędna konwersja układów

C	Obrót działa, translacja nie	Wejście 3DoF; nieaktywna akcja; śledzenie pozy wyłączone
D	Kamera porusza się podwójnie	Dwie aktywne kamery; zagnieżdżone rigi; własny skrypt i provider

Procedura diagnostyki (15 minut):

1. Opisz objaw mierzalnie. Np. „Sześcian ma 0.01 m według Bounds", nie „wygląda dziwnie".
2. Sformułuj co najmniej dwie hipotezy. Każda musi przewidywać inny wynik testu.
3. Wybierz test rozstrzygający. Sprawdź hierarchię, Inspector, log wejścia albo aktywny loader - zgodnie z hipotezą.
4. Wprowadź najmniejszą poprawkę. Nie zmieniaj kilku parametrów jednocześnie.
5. Powtórz pomiar. Zapisz wynik przed i po poprawce.
6. Wykonaj commit. W komunikacie nazwij przyczynę, nie objaw: git add Assets evidence/lab01.md
   git commit -m "fix(lab01): correct <przyczyna-błędu>"

PUNKT KONTROLNY D: Diagnoza zawiera objaw, dwie hipotezy, test rozstrzygający, przyczynę i wynik przed i po poprawce. Commit opisuje przyczynę, nie objaw.

Checkpoint 4.5

Wymagania dodatkowe (oprócz Checkpointów 3.0 + 4.0):

1. Tabela porównawcza: OpenXR vs Meta Quest SDK vs ARCore
2. Dla każdego runtime: DoF, typ śledzenia, dostępne akcje, rozszerzenia opcjonalne
3. Interpretacja praktyczna: ograniczenie → konsekwencja dla aplikacji (np. brak 6DoF → niemożliwe statyczne obiekty otoczenia)
4. Rekomendacja: który runtime wybrać do konkretnego zastosowania (medyczne, przemysłowe, edukacyjne)
5. Wnioski łączące technologię z problemem inżynierskim

Weryfikacja (prowadzący, max 60s):
✓ Checkpointy 3.0 + 4.0: wszystkie kryteria
✓ Tabela porównawcza runtime'ów
✓ Interpretacja ograniczeń: teoria → praktyka
Zaliczenie: 2/3 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.5

Checkpoint 5.0

Wymagania dodatkowe (oprócz checkpointów 3.0, 4.0, 4.5)

1. Propozycja optymalizacji: np. zmiana trybu śledzenia, wykorzystanie rozszerzeń OpenXR, kalibracja względem konkretnego runtime
2. Eksploracja granic: test graniczny - co się dzieje na granicy 6DoF (np. poniżej progu śledzenia), gdzie API zgłasza niedostępność vs fizyczne ograniczenia
3. Zaproponowany scenariusz, w którym audyt hardware'u bezpośrednio wpłynął na projekt (np. ponieważ VITURE nie ma 6DoF, aplikacja użyła innej strategii lokalizacji)
4. Wnioski zawierające lekcję dla następnych projektów XR

Weryfikacja (prowadzący, max 60s):
✓ Checkpointy 3.0 + 4.0 + 4.5: wszystkie kryteria
✓ Propozycja optymalizacji, poparta pomiarami
✓ Eksploracja granic, scenariusz praktyczny, wnioski
Zaliczenie: 2/3 dodatkowe kryteria (+ poprzednie). Ocena finalna: 5.0

Oddanie

Nie przygotowuj osobnego sprawozdania. Uzupełnij plik evidence/lab01.md według poniższego wzoru i wykonaj commit.

```
# Laboratorium 1
- <para / nazwiska>
## Identyfikacja pary
```

```
- Numer indeksu student 1: _____
- Numer indeksu student 2: _____
- Suma numerów: _____
- Przypisany wariant diagnostyczny (ze wzoru): ___ (A/B/C/D)
## Wersje
- Unity / OpenXR / XRI: ___ / ___ / ___
- Platforma i runtime: _____
## Scena i geometria
- Rozmiar Cube_1m: ___ m
- Pozycja Child: przewidywana ___ / zmierzona ___
- Tracking origin, a world origin: _____
## Pomiar bazowy
| Środowisko | Rozdzielczość | 3 × FPS | Mediana FPS | Mediana ms ||---|---|---|---|---|| ___ | ___ | ___, ___, ___ | ___ | ___ |
## Audyt urządzeń
| Urządzenie | DoF | Wejście | API/Runtime | Ograniczenie | Dowód ||---|---|---|---|---|---|| Quest 3 | ___ | ___ | ___ | ___ | ___ || VITURE | ___ | ___ | ___ | ___ | ___ || Android | ___ | ___ | ___ | ___ | ___ || Symulator | ___ | ___ | ___ | ___ | ___ |
## Diagnoza (wariant ___)
- Objaw / hipotezy: ___ / ___
- Test rozstrzygający: ___
- Przyczyna: ___
- Wynik przed / po: ___ / ___
## Wnioski
1. ___
2. ___
3. ___
![Scena bazowa](img/lab01-scene.png)
```

Weryfikacja prowadzącego; maksymalnie 60 sekund na każdy checkpoint

Kryterium	Dowód	Status
Uruchomienie	Play Mode, brak błędów krytycznych	zal. / niezal.
Geometria	Cube_1m = 1m, osie, poprawna hierarchia	zal. / niezal.
Śledzenie	Obrót i translacja w symulatorze	zal. / niezal.
Pomiar	Trzy odczyty, mediana, kontekst	zal. / niezal.
Audyt	Funkcje oddzielone od „niezweryfikowano"	zal. / niezal.
Diagnoza	Test rozstrzygający, poprawka, wkład obu osób	zal. / niezal.

Zasada zaliczenia

Ćwiczenie ma status „zaliczone" (checkpoint 3.0), gdy spełniono kryteria uruchomienia, geometrii i diagnozy oraz co najmniej dwa z trzech pozostałych kryteriów. Zaliczenie stanowi bazę do przejścia na Lab 2.

Pytania kontrolne

1. Dlaczego aktywny runtime OpenXR i OpenXR Plugin w projekcie nie są tym samym elementem systemu?
2. Jak odróżnić brak translacji wynikający z urządzenia 3DoF od nieaktywnej akcji wejścia?
3. Jaki skutek ma skala 0.01 obiektu nadrzędnego dla potomka o lokalnym przesunięciu 1 m?
4. Dlaczego XR Origin nie powinien być bez potrzeby skalowany?
5. Które informacje trzeba zapisać, aby wynik FPS był porównywalny w kolejnych laboratoriach?
6. Dlaczego passthrough widoczny użytkownikowi nie dowodzi, że aplikacja może odczytać obraz kamer?
7. Kiedy właściwą wartością w karcie audytu jest „niezweryfikowano", a kiedy „niedostępne"?

Exit ticket - odpowiedz indywidualnie

Jedno zdanie: Której funkcji badanego sprzętu nie wolno założyć bez sprawdzenia konkretnego API/runtime i jaki dowód byłby wystarczający do jej potwierdzenia?

Materiały rozszerzające (po ukończeniu checkpointu 3.0)

Poniższe zadania można wykonywać równolegle z ulepszaniem checkpointów :

1. Rozszerz rejestrator o zapis co 1 s do pliku CSV i porównaj medianę z 95. percentylem czasu klatki.

2. Zbuduj tę samą scenę jako samodzielną aplikację PC i porównaj wynik z Edytorem przy identycznej rozdzielczości.

3. Napisz test Edit Mode sprawdzający, czy Cube_1m ma skalę (1, 1, 1), a XR Origin nie ma niejednostkowej skali.

4. Dodaj ekran diagnostyczny prezentujący nazwę aktywnego XR loadera i status podsystemów wejścia.

Diagnostyka typowych problemów

Objaw	Sprawdź w tej kolejności	Nie rób
Projekt otwiera się z migracją	ProjectVersion.txt → Unity Hub → gałąź	Nie zapisuj projektu w nowszej wersji
Czerwone błędy po imporcie	Pierwszy błąd Console → pakiety/lock → Input System	Nie kasuj losowo Library/ ani manifestu
Czarny lub pusty Game View	Aktywna kamera → clipping → pozycja XR Origin	Nie dodawaj kolejnej kamery bez diagnozy
Symulator nie reaguje	Focus Game View → prefab symulatora → Input Action Manager	Nie twórz od razu nowych bindings
Ruch jest odwrócony	Hierarchia → obrót rodzica → układ osi → konwersja	Nie kompensuj błędu ujemną skalą
Niski FPS w Edytorze	Profiler GPU/CPU → Game View → narzut Edytora	Nie przedstawiaj jednego odczytu jako benchmarku

Checklista studenta przed oddaniem pracy

☐ Pracuję na wersji Unity zgodnej z ProjectVersion.txt.

☐ Nie zmieniłem wersji pakietów poza zakresem instrukcji.

☐ W scenie istnieje jedna aktywna kamera główna.

☐ XR Origin ma skalę (1, 1, 1).

☐ Obiekt wzorcowy ma wymiary 1 m × 1 m × 1 m.

☐ Symulator pozwala sprawdzić obrót i translację głowy w 6DoF.

☐ Zapisałem trzy odczyty FPS i ms oraz obliczyłem medianę.

☐ Karta audytu rozróżnia „niedostępne" i „niezweryfikowano".

☐ Diagnoza zawiera objaw, dwie hipotezy, test, przyczynę i wynik po poprawce.

☐ Plik evidence/lab01.md i zrzut znajdują się w repozytorium.

☐ Każda osoba w parze wykonała co najmniej jeden merytoryczny commit.

☐ W skrypcie BaselineFrameTime wpisałem poprawnie sumę numerów indeksów pary.

Bibliografia

1. The Khronos Group, „OpenXR 1.1 Specification", wersja 1.1.61.
2. The Khronos Group, „OpenXR - High-performance access to AR and VR".
3. Unity Technologies, „XR Plug-in Management".
4. Unity Technologies, „XR Interaction Toolkit".
5. Unity Technologies, „XR Device Simulator overview".
6. Unity Technologies, „OpenXR Plugin".
7. Meta, „Set up your headset for development".
8. Unity Technologies, „Android environment setup".