LABORATORIUM 3
Rejestracja i kotwice w rozszerzonej rzeczywistości

AR Foundation • ARCore • płaszczyzny • raycast • kotwice • błąd rejestracji

Problem inżynierski

Jak zlokalizować wirtualny model w rzeczywistym świecie, gdy urządzenie musi samodzielnie estymować swoją pozycję, orientację i skalę bezpośrednio z obserwacji otoczenia? Jak zmierzyć błąd rejestracji i zidentyfikować jego przyczyny?

Parametry laboratorium

Parametr	Wartość
Tryb pracy	pary; jeden telefon Android na parę, jedna osoba skanuje
Rezultat	zarejestrowany model 1:1, pomiar błędu rejestracji, kotwice sesyjne
Oddanie	commit + evidence/lab03.md; bez tradycyjnego sprawozdania
Wersja	1.2 • 06 września 2026 r.

System oceniania

Ocena wynika z osiągnięcia kolejnych checkpointów. Każdy checkpoint weryfikuje prowadzący w ciągu maksymalnie 60 sekund (na urządzeniu).

Ocena	Główne wymagania
3.0	Skanowanie → raycast → model 1:1 → kotwica sesyjna
3.5	Rejestracja dwupunktowa (Origin + kierunek X)
4.0	Pomiar błędu rejestracji po ruchu urządzenia (5 pozycji)
4.5	Porównanie metod rejestracji, analiza przyczyn błędu
5.0	Automatyzacja, eksploracja granic ARCore, propozycje optymalizacji

Walidacja: Numer indeksu i wariant diagnostyczny

Każdy student wpisuje swój numer indeksu. Wariant błędu w Części D wynika ze wzoru:

$$\text{wariantDiagnostyczny} = (\text{numer_indeksu_1} + \text{numer_indeksu_2}) \bmod 5$$

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

Wymagania wstępne

Przed zajęciami

- Zaliczone Laboratorium 1 (transformacje, skala, weryfikacja API)
- Telefon Android zgodny z ARCore, przewód USB i zgoda właściciela na instalację kompilacji
- Włączone opcje programistyczne i debugowanie USB
- Unity z modułami Android Build Support, SDK, NDK zgodnymi z wersją kursu
- Repozytorium kursu bez czerwonych błędów w Console
- Znajomość układów współrzędnych, transformacji hierarchicznych, skala 1 jednostka = 1 m

Pakiety wymagane w projekcie

Pakiet	Identyfikator	Rola
AR Foundation	com.unity.xr.arfoundation	Abstrakcja nad ARCore, ARKit, OpenXR
ARCore XR Plugin	com.unity.xr.arcore	Integracja z ARCore na Android
XR Plug-in Management	com.unity.xr.management	Zarządzanie runtime'ami XR
XR Interaction Toolkit	com.unity.xr.interaction.toolkit	XR Origin i wsparcie interakcji
Input System	com.unity.inputsystem	Akcje wejścia (ekran, gesty)

Dodatkowo:
Wydrukowana mapa pomiarowa z trzema polami (50×50 cm) i przedmioty odniesienia (wydruk A4, sześcian papierowy 1×1×1 m albo jego część).

Bezpieczeństwo, prywatność i organizacja

- Podczas skanowania patrzysz naprzemiennie na ekran i otoczenie. Nie chodź tyłem ani z telefonem zasłaniającym widok.
- Trasę pomiarową wyznacz taśmą w wolnym obszarze wolnym od osób postronnych.
- Jedna osoba obsługuje telefon (skanuje), druga obserwuje otoczenie i zapisuje wyniki.
- Nie filmuj osób postronnych. Aplikacja nie powinna zapisywać obrazu kamery ani danych osobowych.
- Nie instaluj aplikacji z prywatnymi kluczami ani kontami. Po zajęciach właściciel telefonu może usunąć kompilację.
- Mapa pomiarowa i znaczniki muszą być płasko przymocowane. Luźna kartka nie jest stabilnym odniesieniem dla ARCore.
- Nie wyłączaj ochrony systemu i nie omijaj komunikatów bezpieczeństwa.

Podstawy teoretyczne

4.1. Układy współrzędnych w mobilnym AR

Platforma (ARCore) estymuje ruch kamery i tworzy układ sesji (Session Space). XR Origin w scenie odwzorowuje współrzędne sesji na świat gry. Pozycja kamery w sesji to urządzenie, które obsługuje aplikacja. Model wirtualny musi być umieszczony w tym samym układzie współrzędnych, aby być zarejestrowany (aligned).

4.2. Detencja płaszczyzn i raycast

ARPlaneManager śledzi płaszczyzny na podstawie analizy obrazu z kamery. Każda płaszczyzna ma AABB (axis-aligned bounding box). Raycast przecina promień z otoczeniem (geometrią, płaszczyznami). To pozwala użytkownikowi wybrać miejsce, gdzie umieścić model.

4.3. Rejestracja dwupunktowa

Punkt O wyznacza początek modelu. Punkt X wyznacza dodatni kierunek osi X. Wektor OX określa orientację i skalę. Gdy model nie pasuje do rzeczywistości, różnica między estymacją ARCore a rzeczywistością to błąd rejestracji.

4.4. Kotwice sesyjne (Session Anchors)

Kotwica sesyjna przytwierdza obiekt do pozycji w świecie rzeczywistym na czas trwania sesji. Gdy urządzenie się porusza, kotwica zmienia swoją pozycję w scenie, aby pozostać w tym samym miejscu na świecie. Kotwice sesyjne NIE są trwałe - zanikają po zamknięciu aplikacji.

4.5. Błąd rejestracji

Błąd rejestracji = rzeczywista pozycja rzeczywistego obiektu − estymowana pozycja modelu wirtualnego. Może wynikać z: niedokładnej kalibracji kamery, niedostatecznej ilości cech wizualnych, ruchu, zmiany oświetlenia, niedokładności sensora IMU.

Plan pracy

Etap	Dowód ukończenia
Miniquiz i kontrola telefonu	Odpowiedzi + ARCore dostępny
Repozytorium, konfiguracja AR	Scena uruchamia się, płaszczyzny widoczne
Skanowanie, raycast, model 1:1	Model umieszczony na płaszczyźnie, rozmiar 1m×1m×1m
Rejestracja dwupunktowa	Punkt O i X zaznaczone, model wyrównany
Kotwica sesyjna i pomiar błędu	Pięć pomiarów błędu rejestracji z różnych pozycji
Diagnoza błędu	Hipoteza, test, przyczyna, commit
Oddanie i exit ticket	Evidence/lab03.md + odpowiedź

Checkpoint 3.0

Wymagania do zaliczenia:

✓ Projekt uruchamia się na telefonie Android bez błędów krytycznych

✓ ARPlaneManager wykrywa płaszczyzny (widoczne na ekranie)

✓ Raycast działa - użytkownik może wskazać punkt na płaszczyźnie

✓ Model 1:1 (1m×1m×1m) umieszczony na wykrytej płaszczyźnie

✓ Model utrzymuje skalę niezależnie od transformacji XR Origin (XR Origin scale = 1,1,1)

✓ Kotwica sesyjna przyczepia model do świata rzeczywistego

✓ Pomiar błędu rejestracji: pięć pozycji, każda z dystansem od początkowego umieszczenia

✓ Diagnoza: objaw, dwie hipotezy, test rozstrzygający

✓ Evidence/lab03.md zawiera: numery indeksów, wyniki pomiarów, wariant diagnostyczny

Część A. Uruchomienie projektu AR

A.1. Miniquiz (10 minut, indywidualnie)

1. Wyjaśnij, dlaczego XR Origin scale musi być (1,1,1) nawet w mobilnym AR.
2. Czym różni się raycast w AR od raycastu w VR (brak Known Pose)?
3. Kiedy ARPlaneManager zwróci pustą listę płaszczyzn i co może to spowodować?
4. Wskaż różnicę między Session Anchor a Cloud Anchor.
5. Podaj dwie przyczyny, dla których model może być nieskalibrowany nawet po umieszczeniu.

A.2. Repozytorium i konfiguracja (10 minut)

1. git clone <ADRES> xr-lab && cd xr-lab
2. git fetch --all --tags && git switch lab03-start
3. git switch -c team-<nr>/lab03-<nazwisko1>-<nazwisko2>
4. Odczytaj ProjectSettings/ProjectVersion.txt - zainstaluj tę wersję
5. Otwórz Assets/Scenes/Lab03_AR.unity w Unity Hub

A.3. Kontrola konfiguracji Android (5 minut)

1. Edit → Project Settings → Player → Android: ARCore Required = false (opcjonalne)
2. Edit → Project Settings → XR Plug-in Management: zaznacz ARCore dla Android
3. Edit → Project Settings → ARCore: sprawdź dostępne rozszerzenia (Depth API, Instant Placement)
4. Window → Pair Manager: sprawdzenie dostępności czterech pakietów AR + XRI

A.4. Kompilacja i uruchomienie na telefonie (10 minut)

1. File → Build Settings → Android; Player: Graphics Vulkan (lub OpenGLES 3 jeśli Vulkan zawiera błędy)
2. File → Build and Run; apk instaluje się na telefonie
3. Po uruchomieniu: pozwól na dostęp do kamery i IMU
4. Przesuń telefon po otoczeniu - powinna pojawić się płaszczyzna (zieleń)

PUNKT KONTROLNY A:
Aplikacja uruchamia się na telefonie, ARCore inicjalizuje się, płaszczyzny są widoczne.

Część B. Skanowanie, raycast i model 1:1

B.1. Skanowanie stanowiska (15 minut)

1. Wyznacz trasę pomiarową taśmą - co najmniej 3m × 3m wolnego obszaru
2. Umieść mapę pomiarową (wydruk) na podłożu
3. Skanuj otoczenie: poruszeń telefon w różnych kierunkach, pionowo i poziomo

4. Dostosuj warunki oświetlenia - ARCore raczej nie lubi silnych cieni

5. Czekaj, aż ARPlaneManager zidentyfikuje co najmniej jedną solidną płaszczyznę

B.2. Raycast i umieszczenie modelu (15 minut)

1. Implementuj skrypt ARRaycastManager.Raycast(screenPos, hits)

2. Po naciśnięciu ekranu: raycast w kierunku kamery z tego punktu

3. Jeśli hit jest trafiony: spraw, by model pojawił się w trafionej pozycji

4. Sprawdź hit.trackable - czy to Plane?

5. Model powinien wyglądać na 1m×1m×1m (zaznacz wymiary kolorami lub siatką)

B.3. Kotwica sesyjna (10 minut)

1. Po umieszczeniu modelu: utwórz ARAnchor w pozycji modelu

2. Przypisz model jako dziecko kotwicy: model.transform.SetParent(anchor.transform)

3. Teraz model będzie się poruszać razem z kotwicą, gdy urządzenie się porusza

4. Sprawdzenie: przesuń telefon - model powinien pozostać w tym samym miejscu na świecie

PUNKT KONTROLNY B:
Model 1:1 umieszczony na płaszczyźnie, przypieczętowany kotwicą sesyjną. Czy model pozostaje w tym samym miejscu, gdy urządzenie się porusza?

Część C. Rejestracja dwupunktowa

C.1. Zaznaczenie punkt O (Origin)

1. Naciśnij ekran - utwórz mały marker (sfera lub kostka) w miejscu raycastu

2. To jest punkt O (początek modelu)

3. Zapamiętaj pozycję światową tego punktu

C.2. Zaznaczenie punktu X (kierunek osi X)

1. Naciśnij ekran ponownie w innym miejscu - utwórz drugi marker

2. To jest punkt X na osi X modelu (co najmniej 1m od O)

3. Wektor OX wyznacza kierunek i skalę osi X

4. Oblicz: direction_X = normalize(X − O), scale = distance(O, X)

C.3. Wyrównanie modelu

1. Ustawić localPosition modelu = O (w przestrzeni sesji)

2. Ustawić rotację modelu tak, aby jego oś X pokrywała się z direction_X

3. Sprawdzić skalę: jeśli model ma lokalną skalę (1,1,1) i osi są 1m, to jest już 1:1

4. Wizualizacja: narysuj linie od O do X na ekranie - muszą pokrywać się z rzeczywistością

PUNKT KONTROLNY C:
Model wyrównany dwupunktowo. Punkt O w początku, oś X w kierunku zaznaczonym. Model odzwierciedla rzeczywistość.

Część D. Pomiar błędu rejestracji

D.1. Protokół pomiarowy (20 minut)

Wyznacz 5 pozycji wzdłuż trasy pomiarowej w odstępach 1m. Z każdej pozycji zmierz błąd rejestracji.

1. Pozycja 1: gdzie umieściliśmy model (referencja)
2. Pozycje 2-5: oddalenie się od modelu na 1m, 2m, 3m, 4m wzdłuż ustalonego kierunku
3. Z każdej pozycji: zmierz dystans między rzeczywistym obiektem a modelem wirtualnym (OK: taśma miernicza)
4. Zapisz: dystans od pozycji 1, błąd rejestracji [cm]
5. Oblicz medianę błędu z pięciu pomiarów

```
// Przykład rejestracji błędu w skrypcie
using UnityEngine;
```

```
public class RegistrationError : MonoBehaviour
{
   int pairIndexSum = 0; // WPISZ: suma numerów indeksów
   public void RecordError(float distanceFromOriginMeters, float errorMeters)
{
      Debug.Log(
         $"REGISTRATION_ERROR distance={distanceFromOriginMeters:F2}
         m error={errorMeters:F2}
         m seed={pairIndexSum}"
      );
   }
}
```

Część E. Zadanie diagnostyczne

Wyznaczenie wariantu:

wariant = (numer_indeksu_1 + numer_indeksu_2) mod 5

Wariant	Objaw	Hipotezy
A	Model obraca się gdy porusza się urządzenie	Błędna rotacja z IMU | Odwrócenie osi | Błąd kalibracji kompasu
B	Model dryfuje (zmienia pozycję) przy stałej pozycji urządzenia	Drag integratora pozycji | Bład VIO (Visual Inertial Odometry) | Plany zmienia się znacznik
C	Raycast nie trafia nawet na widoczną płaszczyznę	ARPlaneManager ma brakujące płaszczyzny | Zduplikowana kamera | Błędny layermask raycastu
D	Model rośnie/maleje w górze na ekranie	Camera clipping | XR Origin źle skalowana | Błęd near/far plane
E	Błąd rejestracji rośnie liniowo ze dystansem	Błąd skalowania ARCore | Niedokładna kalibracja kamery | Dryft IMU

1. Opisz objaw mierzalnie (np. „Model dryftuje 5 cm na sekundę przy stałej urządzenia")
2. Sformułuj dwie hipotezy z różnymi przewidywaniami
3. Wykonaj test rozstrzygający (log, wizualizacja, zmiana parametru)
4. Wprowadź jedną minimalną poprawkę
5. Powtórz pomiar - czy objaw znika?
6. git commit -m "fix(lab03): <przyczyna>"

PUNKT KONTROLNY D:
Diagnoza zawiera objaw, dwie hipotezy, test, przyczynę, wynik. Commit opisuje przyczynę.

Checkpoint 3.5

Wymagania dodatkowe (oprócz checkpointu 3.0)

✓ Rejestracja dwupunktowa: punkt O i punkt X zaznaczone jednoznacznie

✓ Wyrównanie modelu: oś X pokrywa się z rzeczywistością (wizualizacja na ekranie)

✓ Skalę modelu zmierzone i potwierdzone: rzeczywisty wymiar vs wymiar wirtualny

✓ Pięć pomiarów błędu z różnych odległości, mediana obliczona

✓ Interpretacja: czy błąd rośnie ze dystansem czy pozostaje stały?

Weryfikacja (prowadzący, max 60s):
✓ Checkpoint 3.0: wszystkie kryteria
✓ Dwupunktowa rejestracja: punkty O i X zweryfikowane
✓ Wyrównanie: oś X wizualnie zgadza się z rzeczywistością
✓ Pomiar błędu: pięć wartości, mediana, trend

Zaliczenie: 3/4 dodatkowe kryteria (+ Checkpoint 3.0)

Checkpoint 4.0

Wymagania dodatkowe (oprócz checkpointów 3.0 + 3.5)

✓ Analiza błędu rejestracji: tabela z pięcioma pomiarami

✓ Identyfikacja trendu: błąd stały vs liniowy vs kwadratowy rośnie

✓ Hipoteza przyczyny błędu: Visual Inertial Odometry vs kalibracja vs dryft

✓ Diagnoza Części E: przyczyna zidentyfikowana i poprawka wdrożona

✓ Porównanie przed/po poprawką

Weryfikacja (prowadzący, max 60s):
✓ Checkpointy 3.0 + 3.5: wszystkie kryteria
✓ Tabela błędów: trend zidentyfikowany
✓ Diagnoza: hipoteza → test → poprawka
✓ Wynik: błąd zmniejszony?
Zaliczenie: 3/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.0

Checkpoint 4.5

Wymagania dodatkowe (oprócz checkpointów 3.0 + 4.0)

✓ Porównanie trzech metod rejestracji: dwupunktowa vs automatyczna (ARCore) vs manualny offset

✓ Dla każdej metody: pomiar błędu, mediana, stabilność

✓ Analiza warunków: oświetlenie, tekstura otoczenia, ruch urządzenia

✓ Propozycja optymalizacji: ulepszenie metody dwupunktowej na podstawie wyników

✓ Wnioski dla następnych projektów AR

Weryfikacja (prowadzący, max 60s):
✓ Checkpointy 3.0 + 4.0: wszystkie kryteria
✓ Trzy metody: porównanie tabelaryczne
✓ Analiza warunków: oświetlenie, tekstura, ruch
✓ Propozycja optymalizacji z danymi
Zaliczenie: 3/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.5

Checkpoint 5.0

Wymagania dodatkowe (oprócz checkpointów 3.0 + 4.0 + 4.5)

✓ Eksploracja granic ARCore: testowanie w warunkach granicznych (słabe oświetlenie, minimalna tekstura, szybki ruch)

✓ Pomiar błędu w każdym wariancie, analiza kompromisu szybkość ↔ dokładność

✓ Automatyzacja rejestracji: algorytm, który minimalizuje błąd bez interwencji użytkownika

✓ Zaproponowane ulepszenia: zmiana algorytmu, dodatkowe sensory (magnetometr), kalibracja multi-punkt

✓ Wnioski: limitacje ARCore na Android, porównanie z innymi platformami AR

Weryfikacja (prowadzący, max 60s):
✓ Checkpointy 3.0-4.5: wszystkie kryteria
✓ Eksploracja granic: pomiary w warunkach granicznych
✓ Automatyzacja: algorytm + pomiary✓ Propozycje + wnioski
Zaliczenie: 3/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 5.0

Oddanie

```
# Laboratorium 3 - <para / nazwiska>
## Identyfikacja pary
- Numer indeksu student 1: _____
- Numer indeksu student 2: _____
- Suma numerów: _____
- Przypisany wariant diagnostyczny: ___ (A/B/C/D/E)
## Wersje i konfiguracja
- Unity / AR Foundation / ARCore: ___ / ___ / ___
- Model telefonu Android: _____
- Wersja ARCore (z aplikacji): _____
## Rejestracja dwupunktowa
- Punkt O (origin): ___ m, ___ m, ___ m
- Punkt X (kierunek osi X): ___ m, ___ m, ___ m
- Dystans OX: ___ m
- Wyrównanie: czy oś X pokrywa się z rzeczywistością? TAK / NIE
## Pomiar błędu rejestracji
| Pozycja | Dystans od O [m] | Błąd rejestracji [cm] |
|---|---|---|
| 1 | 0 | ___ |
| 2 | 1 | ___ |
| 3 | 2 | ___ |
| 4 | 3 | ___ |
| 5 | 4 | ___ |
- Mediana błędu: ___ cm
- Trend: stały / liniowy / inne _____
## Diagnoza (wariant ___)
- Objaw: ___
- Hipoteza 1: ___ → przewidywanie: ___
- Hipoteza 2: ___ → przewidywanie: ___
- Test rozstrzygający: ___
- Przyczyna: ___
- Wynik przed / po: ___ / ___
## Wnioski
1. ___
2. ___
3. ___
![Model zarejestrowany](img/lab03-registration.png)
```

Każda osoba w parze musi mieć co najmniej jeden merytoryczny commit.

Checklista studenta przed oddaniem pracy

☐ Aplikacja uruchamia się na telefonie Android bez błędów krytycznych.

☐ ARPlaneManager wykrywa płaszczyzny w rzeczywistym otoczeniu.

☐ Raycast działa - można wskazać punkt na płaszczyźnie ekranem.

☐ Model 1m×1m×1m umieszczony na płaszczyźnie.

☐ XR Origin ma skalę (1,1,1).

☐ Kotwica sesyjna przyczepia model do świata rzeczywistego.

☐ Model pozostaje w tym samym miejscu, gdy urządzenie się porusza.

☐ Punkt O i punkt X zaznaczone jednoznacznie.

☐ Oś X modelu pokrywa się z rzeczywistością (wizualizacja na ekranie).

☐ Zapisałem pięć pomiarów błędu rejestracji i obliczyłem medianę.

☐ Diagnoza zawiera objaw, dwie hipotezy, test, przyczynę, wynik.

☐ Evidence/lab03.md i zrzut znajdują się w repozytorium.

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

Pytania kontrolne

1. Dlaczego XR Origin scale musi być (1,1,1) nawet w mobilnym AR?
2. Czym różni się raycast w AR od raycastu w VR (brak Known Pose)?
3. Kiedy ARPlaneManager zwraca pustą listę płaszczyzn i co może to spowodować?
4. Wskaż różnicę między Session Anchor a Cloud Anchor.
5. Które warunki powodują większy błąd rejestracji: oświetlenie czy tekstura otoczenia?
6. Dlaczego model rośnie/maleje przy zmianie odległości od kamery?
7. Jak zidentyfikować, czy błąd pochodzi z ARCore czy z błędu człowieka?

Exit ticket (odpowiedź indywidualnie)

Jedno zdanie: Które ograniczenie ARCore (dostępność sensora, algorytm, czy hardware'u) byłoby najtrudniejsze do przezwyciężenia w Twoim projekcie i dlaczego?