LABORATORIUM 5
Optymalizacja modeli CAD do czasu rzeczywistego

Import CAD • hierarchia • redukcja poligonów • LOD • benchmark • budżet wydajności

Problem inżynierski

Jak przekształcić model CAD (miliony poligonów, optymalizowany dla dokładności inżynierskiej) w efektywny model czasu rzeczywistego przeznaczony dla AR/VR? Jak zmierzyć i zoptymalizować wydajność bez utraty ważnych szczegółów geometrycznych?

Parametry laboratorium

Parametr	Wartość
Czas	120 minut
Tryb pracy	pary; każda para pracuje na własnym modelu CAD
Rezultat	zoptymalizowany model z LOD0/1/2, benchmark, raport budżetu
Oddanie	commit + evidence/lab05.md; bez tradycyjnego sprawozdania
Wersja	1.0 • 18 października 2026 r.

System oceniania

Ocena wynika z osiągnięcia kolejnych checkpointów. Każdy checkpoint weryfikuje prowadzący w ciągu maksymalnie 90 sekund (demonstracja).

Ocena	Główne wymagania
3.0	Import CAD → hierarchia → LOD0 → benchmark wydajności
3.5	LOD1 i LOD2 implementacyjne, rosnąca złożoność vs wydajność
4.0	Benchmark A/B (z LOD / bez LOD), analiza wpływu na FPS
4.5	Audyt budżetu: vertex shader, pixel shader, kolizje, jakość materiałów
5.0	Automatyzacja redukcji, eksploracja granic, propozycje optymalizacji

Walidacja: Numer indeksu i wariant diagnostyczny

Każdy student wpisuje numer indeksu. Wariant błędu 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)

Wymagania wstępne

Przed zajęciami

Zaliczone Laboratorium 1 (transformacje, skala 1m=1 jednostka)

Model CAD dostarczony przez prowadzącego (FBX, GLTF albo 3DS Max/Blender źródło)

Unity z FBX importer (domyślnie zainstalowany)

Repozytorium kursu bez czerwonych błędów

Profiler Unity (Window → Analysis → Profiler)

Znajomość LOD (Level of Detail), kolizji, materiałów

Modele wejściowe

Model	Poligony (CAD)	Wersja optymalna

Workbench Industrial	~450k	LOD0: 40k, LOD1: 15k, LOD2: 5k
Robot Assembly	~2.8M	LOD0: 200k, LOD1: 80k, LOD2: 20k
Machine Part	~180k	LOD0: 18k, LOD1: 7k, LOD2: 2k

Każda para otrzymuje jeden model. Warianty (A/B/C/D) określają, którą część stanowiska optymalizować.

Bezpieczeństwo i integralność danych

Nie modyfikuj oryginalnego pliku CAD. Zawsze pracuj na kopii w Assets/Models/

Historia Git: każdy commit powinien reprezentować logiczny krok (import, hierarchia, LOD, benchmark)

Raporty pomiarowe oznacz datą i czasem — powtarzalność jest ważna.

Nie usuwaj oryginalnych poligonów przed zmierzeniem benchmarku baseline'u.

Kolaboracja: jedna osoba optymalizuje geometrię, druga weryfikuje benchmark.

Podstawy teoretyczne

4.1. CAD vs model czasu rzeczywistego

Model CAD jest optymalizowany dla dokładności: siatka jest gęsta, aby reprezentować krzywe i powierzchnie. Model czasu rzeczywistego jest optymalizowany dla wydajności: mniej poligonów, mniej textury, prostsze materiały. Konwersja nie jest automatyczna — wymaga zrozumienia celu użytkownika (edukacja, konserwacja, montaż).

4.2. Jednostki, osie i transformacje

CAD zwykle wykorzystuje cm albo mm. Unity stosuje 1 jednostka = 1 metr. Import FBX może automatycznie skalować: sprawdź Import Settings → Model → Scale Factor. Osie mogą się nie pokrywać (X w prawo w CAD vs Z w prawo w Unity). Zawsze sprawdzaj transformację root'a przed zafixowaniem LOD.

4.3. Budżet wydajności

Budżet = maksymalna liczba poligonów/materiałów/textury na scenę, aby uzyskać docelowy FPS. Dla 90 Hz (11.11 ms budżet): GPU może przetworzyć ~500k–1M poligonów w zależności od złożoności shadera. LOD zmniejsza poligony gdy model jest daleko od kamery.

4.4. LOD (Level of Detail)

LOD0 (bliska): pełna geometria. LOD1 (średnia): 50% poligonów. LOD2 (daleka): 10–20% poligonów. Unity automatycznie przełącza meshes na podstawie odległości. Każdy LOD powinien być wizualnie poprawny bez artefaktów.

4.5. Kolider

Mesh Collider konwertuje geometrię na kolider fizyki. Dla złożonych modeli jest drogi. Alternatywa: proste kształty (Box, Sphere, Capsule) albo ręcznie narysowane uproszczone kolider.

Plan pracy

Etap	Dowód ukończenia
Wejściówka i predykcja	Odpowiedzi indywidualne
Import i hierarchia	Model widoczny w scenie, struktura sprawdzona
Benchmark baseline (CAD)	Profiler: 3 pomiary FPS, mediana
Redukcja LOD0 i LOD1	Dwa poziomy detalu, wymiary zmierzone
LOD2 i benchmark A/B	Trzy LOD konfiguracyjne, porównanie FPS
Analiza budżetu i diagnoza	Raport: poligony, materiały, kolizje, błąd
Oddanie i exit ticket	Evidence/lab05.md + odpowiedź indywidualna

Checkpoint 3.0

Wymagania do zaliczenia:

✓ Model CAD zaimportowany do Unity i widoczny w scenie

✓ Hierarchia złożona z rozumnych partów (nie zawalonych w jeden obiekt)

✓ Skala sprawdzona: model odpowiada rzeczywistym wymiarom (1m = 1 jednostka)

✓ Benchmark baseline: pomiary FPS dla oryginalnego CAD (3 próby, mediana)

✓ LOD0 zredukowany do 50% oryginalnych poligonów, wciąż pełny szczegół

✓ LOD1 zredukowany do 30% oryginalnych poligonów

✓ Benchmark A/B: pomiary FPS z LOD vs bez LOD

✓ Analiza: gdzie przychodzi główny koszt? (geometria, materiały, kolizje)

✓ Diagnoza: objaw, dwie hipotezy, test rozstrzygający

✓ Evidence/lab05.md zawiera: numery indeksów, wyniki pomiarów, wariant diagnostyczny

Część A. Import CAD i hierarchia

A.1. Wejściówka (10 minut, indywidualnie)

1. Wyjaśnij, dlaczego CAD nie jest gotowy do użytku czasu rzeczywistego.
2. Kiedy LOD2 mogą mieć zerową geometrię zamiast uproszczonej?
3. Które są droższą operacją: renderowanie 100k poligonów czy 10k z złożonym shaderem?
4. Kiedy nie powinno się używać Mesh Collider i jaka jest alternatywa?
5. Co należy sprawdzić PRZED redukcją geometrii?

A.2. Import modelu CAD (10 minut)

1. git clone/switch na gałąź Lab 5 i utwórz gałąź team-<nr>/lab05-<nazwiska>
2. Umieść model CAD w Assets/Models/ (lub Assets/CAD/)
3. Unity: Import FBX — sprawdź Import Settings:
4. - Scale Factor: jeśli CAD jest w mm, zmień na 0.001; jeśli w cm, zmień na 0.01
5. - Rig: Off (nie potrzebujemy animacji)
6. - Materials: zaznacz, aby zaimprtować materiały
7. Dodaj model do sceny; uruchom Play Mode — model powinien być widoczny

A.3. Sprawdzenie hierarchii (10 minut)

1. Zbadaj Hierarchy: czy model ma sensowną strukturę (Parts, Assemblies)?
2. Jeśli model to jeden obiekt: musisz go podzielić ręcznie albo zaimportować ponownie z innym ustawieniem
3. Sprawdź scale root'a: powinien być (1,1,1), a skalowanie powinno być w Import Settings
4. Sprawdzenie wymiarów: zmierz odległość między dwiema widocznymi krawędziami (np. Bounds.size)

PUNKT KONTROLNY A: Model zaimportowany, hierarchia rozsądna, skala (1,1,1) w root'u. Wymiary odpowiadają rzeczywistości.

Część B. Pomiar wydajności baseline'u

B.1. Benchmark oryginalnego CAD (15 minut)

```
// Skrypt do pomiaru wydajnościusing UnityEngine;public class PerformanceBenchmark : MonoBehaviour { int pairIndexSum = 0; // WPISZ: suma numerów indeksów    float frameSum = 0;    int frameCount = 0; void Update() {        frameSum += Time.deltaTime;        frameCount++;                if (frameCount == 300) { // 300 klatek            float avgMs = (frameSum / frameCount) * 1000f;            float fps = 1f / (frameSum / frameCount);            Debug.Log($"BENCHMARK_RAW fps={fps:F1} ms={avgMs:F2} polygons=??? seed={pairIndexSum}");            frameCount = 0;            frameSum = 0;        }    }}
```

1. Wpisz sumę numerów indeksów
2. Uruchom Play Mode przez ~30 sekund bez zmian widoku
3. Zanotuj FPS z trzech pomiarów (każdy po 300 klatkach)
4. Oblicz medianę
5. Profiler → Statistics: ile Draw Calls, ile vertex/fragment shader invocations?

PUNKT KONTROLNY B: Benchmark baseline'u zmierzony. 3 próby, mediana, Draw Calls, shader invocations.

Część C. Redukcja geometrii i LOD

C.1. LOD0 — redukcja do ~50% (20 minut)

1. Otwórz model w Blenderze/3DS Maxie (jeśli dostępny)
2. Użyj Remesher (Blender) albo ProOptimize (3DS Max) do redukcji
3. Cel: 50% poligonów, zachowanie głównych kształtów
4. Eksportuj jako FBX do Assets/Models/model_LOD0.fbx
5. Przypisz w Inspektorze jako LOD0 dla modelu

C.2. LOD1 — redukcja do ~30% (15 minut)

1. Dalsze zmniejszenie: 30% oryginalnych poligonów
2. Możliwe usunięcie małych detali (śruby, kabel)
3. Eksportuj jako model_LOD1.fbx

C.3. Konfiguracja LOD w Unity (10 minut)

1. Dodaj komponent LOD Group do root'a modelu
2. Przypisz meshe: LOD0 (bliska), LOD1 (średnia), LOD2 (daleka)
3. Ustaw progi przejścia (domyślnie: LOD0 0–0.6, LOD1 0.6–0.8)
4. Test: przesuń kamerę — mesh powinien się zmieniać

PUNKT KONTROLNY C: LOD Group konfiguracyjna z trzema poziomami. Przejście między LOD widoczne i bezproblemowe.

Część D. Benchmark A/B i analiza

D.1. Pomiar z LOD (15 minut)

1. Uruchom grę ze skonfigurowanym LOD Group
2. Powtórz pomiar FPS (3 próby, 300 klatek każda)
3. Zanotuj Draw Calls z Profilera
4. Oblicz medianę FPS

D.2. Porównanie A/B (10 minut)

```
// Tabela porównawczaWariant      | Poligony | FPS (mediana) | ms/klatka | Draw CallsBaseline (CAD) | 450k     |
___          | ___      | ___LOD0          | 225k     | ___          | ___      | ___LOD1          | 135k     | ___          | __ _     |
___LOD2 (jeśli) | 45k      | ___          | ___      | ___
```

1. Oblicz przyrost wydajności: (FPS_LOD - FPS_baseline) / FPS_baseline × 100%
2. Gdzie przychodzi największy zysk? LOD0 vs LOD1 vs LOD2?
3. Czy zmiana wizualna jest akceptowalna na dystansach LOD?

PUNKT KONTROLNY D: Tabela A/B wypełniona. Porównanie FPS, analiza wpływu LOD.

Część E. Zadanie diagnostyczne

Wyznaczenie wariantu:

wariant = (numer_indeksu_1 + numer_indeksu_2) mod 4

Wariant	Objaw	Hipotezy
A	LOD przełącza się zbyt wcześnie (bliska kamera, już widoczne LOD1)	Progi LOD Group źle ustawione | Kamera ma zły FOV | zmiana skali world
B	Artefakty na przejściu LOD (wspólne krawędzie się nie zgadzają)	Werteksy nie pokrywają się | UV mapping rozbity | normal maps brakuje
C	Benchmark LOD nie pokazuje przyrostu (FPS taki sam)	LOD Group nie aktywna | Draw Calls nie zmieniają się | shader cost dominuje

D	Model jest cały czarny lub brakuje materiałów w LOD1/2	Materiały nie przypisane | UV coordsy poza zakresem | brakuje textury

1. Opisz objaw mierzalnie (np. „LOD przełącza się przy distancji 2m zamiast 5m")
2. Sformułuj dwie hipotezy z różnymi przewidywaniami
3. Wykonaj test: zmień parametr, obserwuj wynik, log
4. Popraw jedną rzecz; powtórz pomiar
5. git commit -m "fix(lab05): <przyczyna>"

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

Checkpoint 3.5

Wymagania dodatkowe (oprócz Checkpointu 3.0):

✓ LOD2 zredukowany do ~15–20% oryginalnych poligonów

✓ Wszystkie trzy LOD (0, 1, 2) skonfigurowane w LOD Group

✓ Przejścia między LOD są wizualnie gładkie i bez artefaktów

✓ Porównanie wizualne: LOD0 vs LOD2 (czy główne kształty są zachowane?)

✓ Tabela: wymiary geometrii dla każdego LOD (Bounds.size)

Weryfikacja (prowadzący, max 90s demonstracja):✓ Checkpoint 3.0: wszystkie kryteria✓ Trzy LOD konfiguracyjne✓ Przejścia gładkie✓ Wymiary zachowywaneZaliczenie: 2/4 dodatkowe kryteria (+ Checkpoint 3.0)

Checkpoint 4.0

Wymagania dodatkowe (oprócz Checkpointów 3.0 + 3.5):

✓ Audyt budżetu: poligony, materiały, Draw Calls, shader pass'y dla każdego LOD

✓ Analiza: gdzie przychodzi dominujący koszt (vertex vs pixel shader)?

✓ Porównanie Draw Calls: czy redukcja poligonów zmniejsza Draw Calls?

✓ Kolider: czy jest Mesh Collider? Czy może być uproszczony (Box/Sphere)?

✓ Wniosek: który LOD jest optymalny dla docelowego kontekstu (AR/VR, desktop)?

Weryfikacja (prowadzący, max 90s):✓ Checkpointy 3.0 + 3.5: wszystkie kryteria✓ Audyt budżetu: tabela poligony/Draw Calls/shader✓ Analiza bottleneck'u✓ Kolider: propozycja optymalizacjiZaliczenie: 2/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.0

Checkpoint 4.5

Wymagania dodatkowe (oprócz Checkpointów 3.0 + 4.0):

✓ Porównanie wyników z innym modelem CAD (jeśli dostępny)

✓ Skalowalność: czy metoda redukcji sprawdza się dla większych modeli?

✓ Optymalizacja materiałów: czy tekstury mogą być zmniejszone dla LOD1/2?

✓ Propozycja architekury: jak zintegrować LOD w systemie produkcyjnym?

✓ Wnioski: limitacje metody, przyszłe kierunki automatyzacji

Weryfikacja (prowadzący, max 90s):✓ Checkpointy 3.0–4.0: wszystkie kryteria✓ Porównanie z innym modelem✓ Analiza skalowania✓ Optymalizacja materiałówZaliczenie: 2/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 4.5

Checkpoint 5.0

Wymagania dodatkowe (oprócz Checkpointów 3.0 + 4.0 + 4.5):

✓ Automatyzacja redukcji: script/tool, który generuje LOD0/1/2 bez interwencji ręcznej

✓ Eksploracja granic: najdramatyczniejsza redukcja możliwa bez utraty funkcjonalności

✓ Pomiary wydajności: FPS, Draw Calls, memory footprint dla każdego LOD

✓ Zaproponowana procedualizacja: LOD dla dowolnego modelu CAD

✓ Wnioski: porównanie z innymi narzędziami (Simplygon, Meshlab), rzeczywiste zastosowania (bliźniaki cyfrowe, AR/VR)

Weryfikacja (prowadzący, max 90s):✓ Checkpointy 3.0–4.5: wszystkie kryteria✓ Automatyzacja: script działa✓ Test na nowym modelu CAD✓ Pomiary memory/FPS/Draw CallsZaliczenie: 2/4 dodatkowe kryteria (+ poprzednie). Ocena finalna: 5.0

Oddanie

Laboratorium 5 — <para / nazwiska>## Identyfikacja pary- Numer indeksu student 1: _____- Numer indeksu student 2: _____- Suma numerów: _____- Przypisany wariant diagnostyczny: ___ (A/B/C/D)## Model CAD i import- Nazwa modelu: _____- Rozmiar oryginału: ___k poligonów- Skala: Factor = _____- Root scale: (__, __, __)## Benchmark baseline| Próba | FPS | ms/klatka | Draw Calls ||---|---|---|---|| 1 | __ | __ | __ || 2 | __ | __ | __ || 3 | __ | __ | __ |- Mediana: __ FPS, __ ms## LOD Hierarchy| Level | Poligony | % oryginału | FPS | ms | Draw Calls ||---|---|---|---|---|---|| LOD0 | __ | __ | __ | __ | __ || LOD1 | __ | __ | __ | __ | __ || LOD2 | __ | __ | __ | __ | __ |## Analiza budżetu- Dominujący koszt: geometria / materiały / shader / kolizje- Vertex shader cost: ____- Pixel shader cost: ____- Mesh Collider: TAK / NIE (jeśli TAK, propozycja optymalizacji: ___)## Diagnoza (wariant ___)- Objaw: ___- Hipoteza 1: ___- Hipoteza 2: ___- Test rozstrzygający: ___- Przyczyna: ___- Wynik przed / po: ___## Wnioski1. ___2. ___![Model baseline vs LOD](img/lab05-lod-comparison.png)

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

Lista końcowa studenta — przed oddaniem

☐ Model CAD zaimportowany do Unity ze skalą (1,1,1).

☐ Hierarchia składa się z rozsądnych części.

☐ Wymiary modelu odpowiadają rzeczywistości.

☐ Benchmark baseline'u zmierzony (3 próby, mediana).

☐ LOD0 i LOD1 zredukowane do ~50% i ~30% oryginalnych poligonów.

☐ LOD Group skonfigurowana w Inspektorze.

☐ Przejścia między LOD są gładkie i bez artefaktów.

☐ Benchmark A/B: pomiary FPS z LOD vs bez LOD.

☐ Audyt budżetu: poligony, Draw Calls, shader cost.

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

☐ Evidence/lab05.md i zrzut znajdują się w repozytorium.

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

Pytania kontrolne

1. Dlaczego CAD nie jest gotowy do czasu rzeczywistego?
2. Jaką skalę ustawić w Import Settings dla modelu w mm?
3. Kiedy LOD1 mogą mieć 50% poligonów zamiast 30%?
4. Które są tańsze: reducja poligonów czy reducja Draw Calls?
5. Czy Mesh Collider powinien używać tego samego mesh co render?
6. Co się dzieje, gdy LOD Group jest wyłączona?
7. Jak zmierzyć wpływ shader'u na wydajność niezależnie od poligonów?

Exit ticket (odpowiedź indywidualnie)

Jedno zdanie: Które ograniczenie optymalizacji CAD (czasu pracy, jakości wizualnej, złożoności algoritmu) byłoby najtrudniejsze w Twoim przyszłym projekcie i jak byś to chciał rozwiązać?