Gra w kości w WPF

Czas: 2 × 45 min | praca indywidualna | Wykład 1.22–1.29

Cel: Zbudujesz aplikację okienkową gry w kości w stylu części desktopowej egzaminu INF.04, wykorzystując klasę `Kosc` z zadania 15 i obrazy kości.

Po zajęciach potrafisz:

- łączy klasę logiki z interfejsem WPF
- wyświetla i zmienia obrazy w kontrolce Image (Build Action: Resource)
- reaguje na kliknięcie obrazu i sygnalizuje stan przezroczystością
- kontroluje dostępność przycisków zależnie od stanu gry

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0

Przygotowanie przed zajęciami

- Przygotuj klasę `Kosc` z zadania 15 i pobierz obrazy `kosc0.png` – `kosc6.png` z folderu `pliki` przy tym zadaniu.

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.
- Rób przerwy dla oczu (zasada 20-20-20) — praca z projektantem okien wymaga długiego patrzenia w ekran.

Przebieg zadania

1 Projekt i zasoby

Utwórz projekt WPF `GraWKosci`.

1. Dodaj plik `Kosc.cs` z zadania 15.
2. Utwórz folder `Images`, dodaj obrazy i ustaw im **Akcja kompilacji: Resource**.
3. `kosc0.png` to kość przed rzutem (znak zapytania).

2 Okno

Układ okna według opisu.

1. Nagłówek i krótka instrukcja gry.
2. `UniformGrid` z pięcioma kontrolkami Image (`Img0`…`Img4`, Tag = numer kości).
3. Wiersz z wynikami: punkty w tym rzucie, wynik gry, numer rzutu („Rzut 2 z 3”).
4. Przyciski: Rzuć kośćmi, Zapisz punkty i nowa tura (nieaktywny na początku), Nowa gra.

3 Logika gry

Kod okna.

```
private void Odswiez()
{
    for (int i = 0; i < _kosci.Length; i++)
    {
        _obrazy[i].Source = new BitmapImage(
            new Uri($"Images/kosc{_kosci[i].Wartosc}.png", UriKind.Relative));
        _obrazy[i].Opacity = _kosci[i].Zablokowana ? 0.4 : 1.0;
    }
    // ... punkty, wynik, numer rzutu, IsEnabled przycisków
}
```

1. Tablica pięciu obiektów `Kosc` i tablica obrazów. W tytule okna pokaż `Kosc.LiczbaKosci`.
2. Rzuć: rzut wszystkimi kośćmi (zablokowane się nie zmieniają), zwiększ licznik rzutów w turze; po 3 rzutach przycisk nieaktywny.
3. Kliknięcie obrazu (`MouseLeftButtonDown`) blokuje lub odblokowuje kość — zablokowana ma `Opacity = 0.4`. Przed pierwszym rzutem kliknięcie nic nie robi.
4. Zapisz punkty: dodaj `Kosc.Punkty(...)` do wyniku gry, odblokuj kości, wyzeruj licznik rzutów.
5. Nowa gra: jeśli wynik > 0 — zapytaj o potwierdzenie.

4 Dokumentacja jak na egzaminie

Przygotuj materiały do oceny.

1. Komentarze `///` w kodzie okna i klasy Kosc.
2. Zrzuty: początek gry, kości po rzucie z dwiema zablokowanymi, wynik po kilku turach.
3. Zagraj 5 tur i zapisz wyniki w tabeli. Zatwierdź pracę w Git.

W sprawozdaniu umieść

- ☐ kod okna (XAML i C#)
- ☐ trzy zrzuty z gry
- ☐ tabela 5 tur
- ☐ opis, jak klasa Kosc została użyta bez zmian

Pytania kontrolne

1. Dlaczego obrazy mają Akcję kompilacji Resource?
2. Jak program pilnuje, że w turze są najwyżej 3 rzuty?
3. Co by trzeba zmienić, aby gra miała 6 kości?
4. Dlaczego logikę punktacji można było przetestować bez uruchamiania okna?

Kryteria oceny

Kryterium	Punkty
Zasoby i układ okna	4
Rzut i wyświetlanie obrazów	4
Blokowanie kości z przezroczystością	4
Tury, wynik gry, dostępność przycisków	4
Dokumentacja i zrzuty	4
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).