# PROGRAMOWANIE GIER

FIZYKA, STEROWANIE, TEREN

## CEL LABORATORIUM

Celem zajęć jest zapoznanie się z podstawami obsługi silnika Unity i programowaniem gier. Podczas zajęć zaprezentowane zostaną elementy interfejsu graficznego silnika Unity, a także podstawowa funkcjonalność – ładowanie zewnętrznych zasobów, zmienianie ich parametrów i tworzenie prefabrykatów.

## MATERIAŁY POMOCNICZE

Strona główna silnika Unity: https://unity.com
Pobieranie: https://unity.com/download
Dokumentacja: https://docs.unity3d.com/Manual/index.html
Pakiet: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.7/manual/index.html
Sklep z zasobami: https://assetstore.unity.com

## ZADANIA WYMAGANE DO WYKONANIA W TRAKCIE ZAJĘĆ

Następujące zadania zrealizuj wraz z pozostałymi osobami należącymi do swojego zespołu.

### ZADANIE 1

Zaproponuj elementy fizyki, jakie będą obecne w grze, którą wraz z zespołem tworzycie. Jakie siły zewnętrzne będą obecne? Celem będzie odzwierciedlenie praw świata rzeczywistego czy radosna twórczość? Fizyka będzie głównym elementem gry czy zaledwie środkiem przekazu? Wykorzystacie silnik fizyki trójwymiarowej czy dwuwymiarowej?

### ZADANIE 2

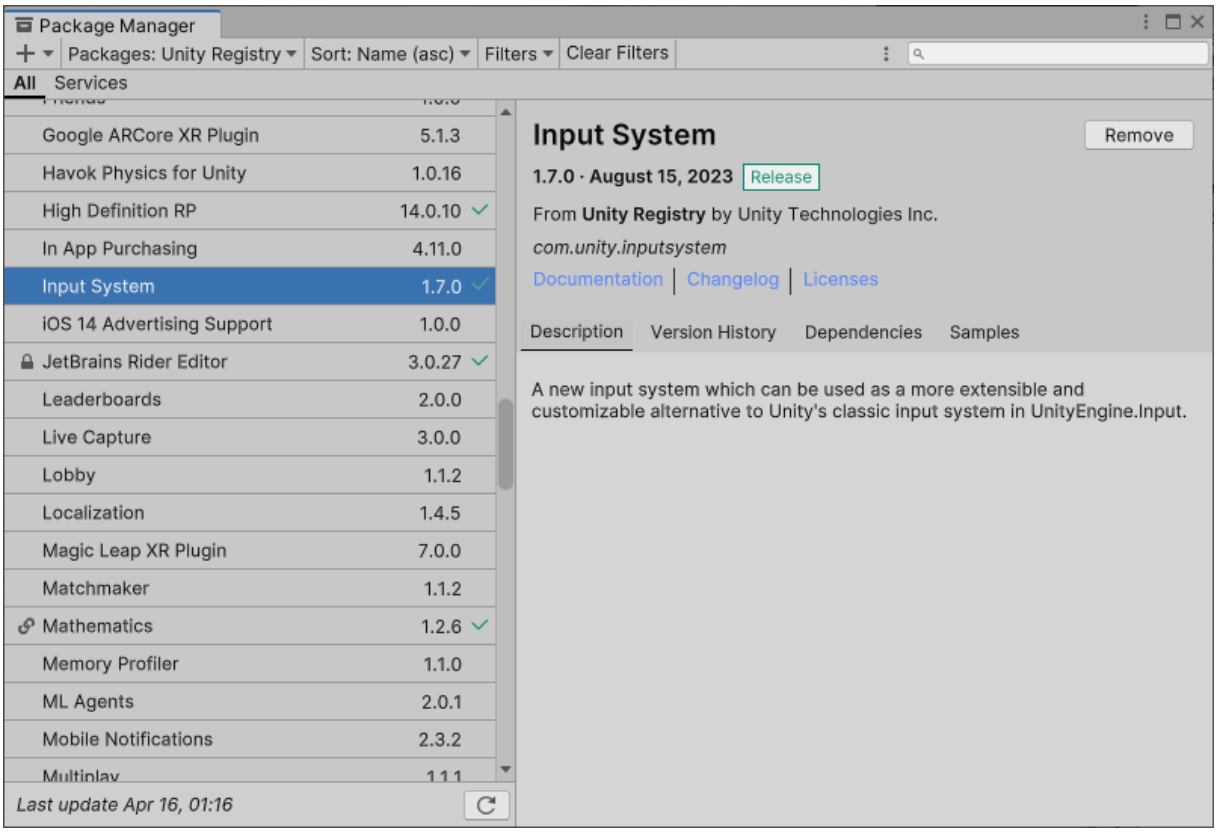
Zaproponuj przestrzeń akcji dostępnych w tworzonej z zespołem grze. Jak będzie się odbywać sterowanie? Jakie urządzenia zostaną wykorzystane?

## ZADANIA DO SAMODZIELNEGO WYKONANIA

Przeczytaj treść następujących zadań. Jeżeli masz podstawowe doświadczenie z przedstawionymi w nich zagadnieniami, możesz je pominąć. W takim wypadku porozmawiaj z zespołem i wykorzystaj czas zajęć, aby wprowadzić nowe elementy do swojej gry.

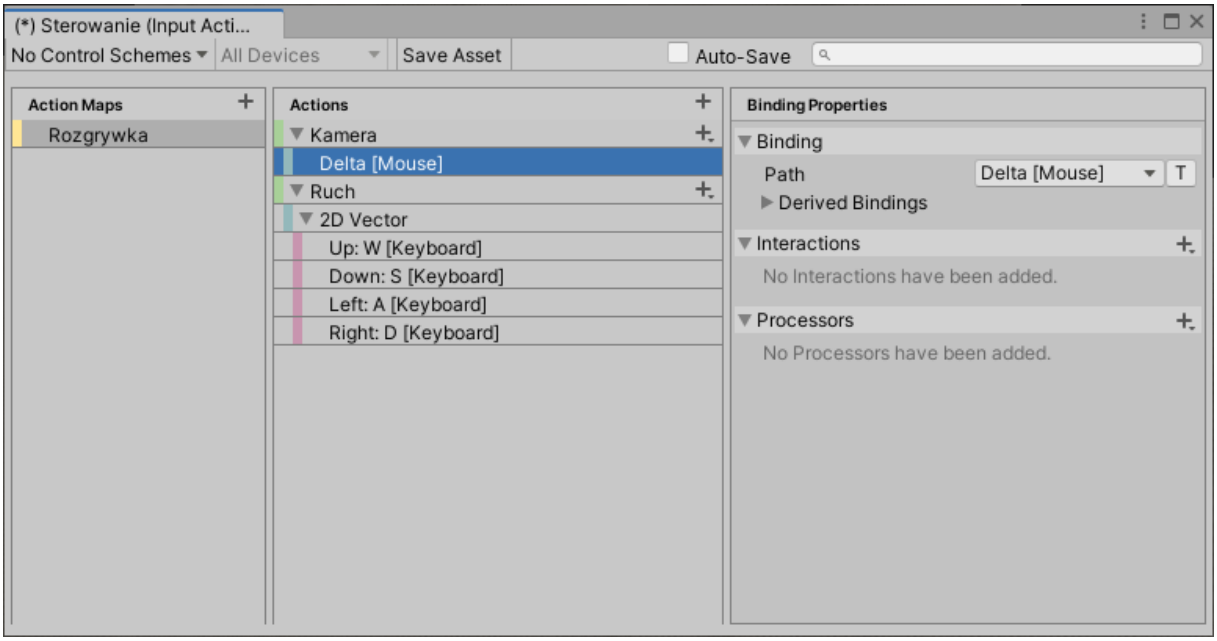
### ZADANIE 3

Uruchom zarządcę pakietów i zainstaluj pakiet Input System w wersji 1.7.0.




# ZADANIE 4

Utwórz nowy zasób typu „Input Actions". Skonfiguruj go w następujący sposób:




# ZADANIE 5

Utwórz nowy skrypt o nazwie „CameraMovement". Otwórz edytor kodu i przepisz do treści skryptu następujący kod. Następnie dodaj komponent skryptu do bytu zawierającego komponent Camera:

```
using UnityEngine;
using UnityEngine.InputSystem; // Krok 1: uzycie pakietu
using UnityEngine.InputSystem.Controls;

public class CameraMovement : MonoBehaviour
{
    public InputActionAsset actions; // Krok 2: referencja do zasobu
    private InputAction akcjaRuchu; // Krok 3: zmienne pomocnicze
    public float predkosc = 100f;

    void Start()
    {
        // Krok 4: wyszukanie referencji do przypisania
        akcjaRuchu = actions.FindActionMap("Rozgrywka").FindAction("Ruch");

        // Krok 5: aktywacja przestrzeni akcji
        actions.FindActionMap("Rozgrywka").Enable();
    }
    void Update()
    {
        // Krok 6: pobranie danych
        Vector2 wektorRuchu = akcjaRuchu.ReadValue<Vector2>();

        // Krok 7: ruch kamery
        Vector3 zmiana = new Vector3(wektorRuchu.x, 0, wektorRuchu.y);
        zmiana *= Time.deltaTime * predkosc;
        transform.position += zmiana;
    }
}
```

Przeanalizuj treść kodu. Jeżeli masz wątpliwości co do jego działania, skonsultuj się z dokumentacją, wykładem dotyczącym systemu wejścia lub prowadzącym.

## ZADANIE 6

Utwórz nowy skrypt o nazwie „CameraRotation". Otwórz edytor kodu i przepisz do treści skryptu następujący kod. Następnie dodaj komponent skryptu do bytu zawierającego komponent Camera:

```
using UnityEngine;
using UnityEngine.InputSystem; // Krok 1: uzycie pakietu

public class CameraRotation : MonoBehaviour
{
    public InputActionAsset actions; // Krok 2: referencja do zasobu
    private InputAction akcjaKamery; // Krok 3: zmienne pomocnicze
    public float predkosc = 5f;

    void Start()
    {
        // Krok 4: wyszukanie referencji do przypisania
        akcjaKamery = actions.FindActionMap("Rozgrywka").FindAction("Kamera");

        // Krok 5: aktywacja przestrzeni akcji
        actions.FindActionMap("Rozgrywka").Enable();
    }
    void Update()
    {
        // Krok 6: pobranie danych
        Vector2 wektorObrotu = akcjaKamery.ReadValue<Vector2>();
```

```
        // Krok 7: ruch kamery
        Vector3 zmiana = new Vector3(wektorObrotu.y, wektorObrotu.x, 0);
        zmiana *= Time.deltaTime * predkosc;
        transform.eulerAngles += zmiana;
    }
}
```

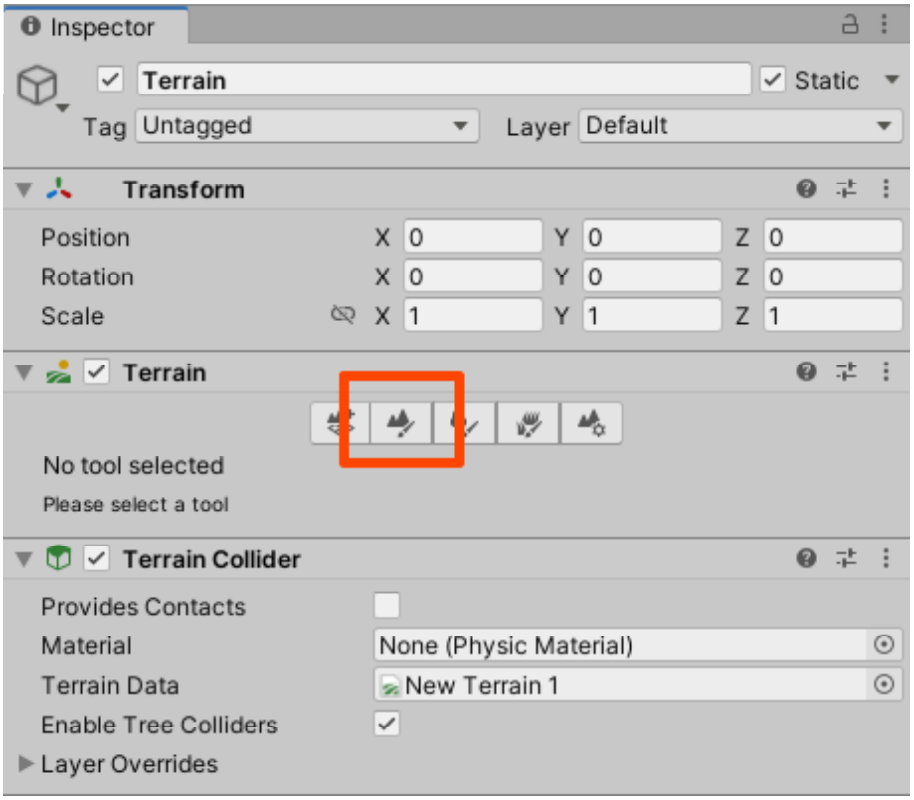
Przeanalizuj treść kodu. Jeżeli masz wątpliwości co do jego działania, skonsultuj się z dokumentacją, wykładem dotyczącym systemu wejścia lub prowadzącym.

# ZADANIE 7

Utwórz kilka różnych bytów na scenie i umieść je w takich miejscach, by były widoczne przez kamerę sceny, a następnie uruchom tryb symulacji. Poruszaj myszką i zaobserwuj efekt. Przytrzymaj dowolny z klawiszy 'W', 'S', 'A' lub 'D' na klawiaturze i zaobserwuj efekt. Wybierz byt kamery i do skryptów, które zostały dodane w zadaniach 5 i 6, przekaż referencję do zasobu konfiguracji sterowania, który został utworzony w zadaniu 4. Ponownie uruchom tryb symulacji i zaobserwuj w jaki sposób uległo zmianie działanie myszy i klawiatury.

# ZADANIE 8

Utwórz byt typu Terrain. Zaznacz go i w oknie inspektora kliknij ikonkę „Paint Terrain".



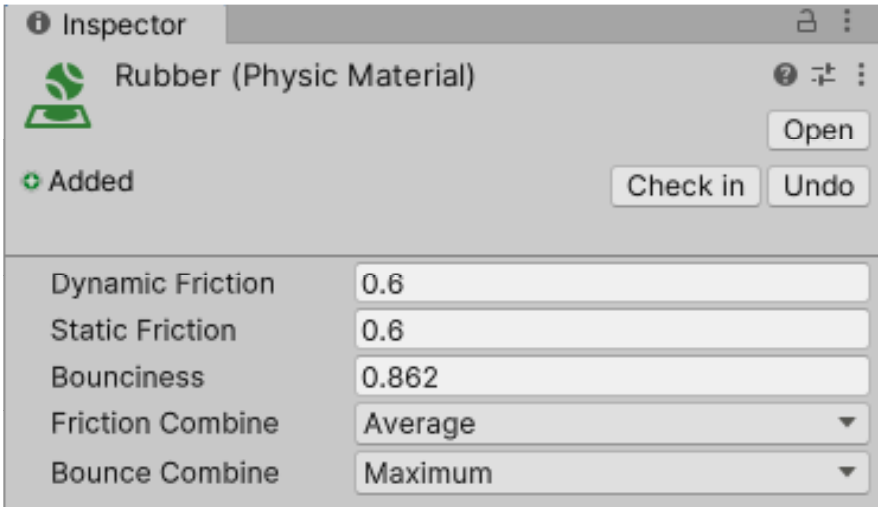

Z listy rozwijanej wybierz tryb „Raise or Lower Terrain". *Brush size* ustaw na wartość 700, a *Opacity* na 70. Upewnij się, że jesteś w trybie edycji sceny i, że kamera jest ustawiona w taki sposób, że widzisz cały obiekt Terrain. Ustaw kursor nad obiektem Terrain, wciśnij i przytrzymaj lewy przycisk myszki. Wypróbuj inne tryby, pędzle i wartości parametrów.

# ZADANIE 9

Utwórz byt typu Sphere i umieść go nad środkiem obiektu Terrain, który został utworzony w zadaniu 8. Uruchom tryb symulacji i zaobserwuj zachowanie kuli. Wróć do trybu edycji sceny i dodaj do kuli komponent RigidBody. Ponownie uruchom tryb symulacji i zaobserwuj działanie kuli.

# ZADANIE 10

Utwórz zasób typu „Physic Material". Przypisz go do kuli utworzonej w zadaniu 9. Ustaw jego wartości w sposób pokazany na poniższym zrzucie ekranu. Uruchom tryb symulacji i zaobserwuj działanie kuli.




# ZADANIE 11

Utwórz nowy skrypt o nazwie „CollisionDetection". Otwórz edytor kodu i przepisz do treści skryptu następujący kod. Następnie dodaj komponent skryptu do bytu kuli, który został utworzony w zadaniu 9:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollisionDetection : MonoBehaviour
{
    private void OnCollisionEnter(Collision collision)
    {
        GameObject bytKolidujacy = collision.gameObject;
        string nazwaObiektu = bytKolidujacy.name;
        ContactPoint miejsceZderzenia = collision.GetContact(0);
        float energia = miejsceZderzenia.impulse.magnitude;
        Debug.Log(string.Format("Kontakt z {0} w punkcie {1} z siłą {2}!",
            nazwaObiektu,
            miejsceZderzenia.point,
            energia)
            );
    }
```

```
    private void OnCollisionStay(Collision collision)
    {
        Debug.Log("Utrzymuje kontakt.");
    }

    private void OnCollisionExit(Collision collision)
    {
        Debug.Log("Odbite.");
    }
}
```

Uruchom tryb symulacji. Obserwuj konsolę Unity.