

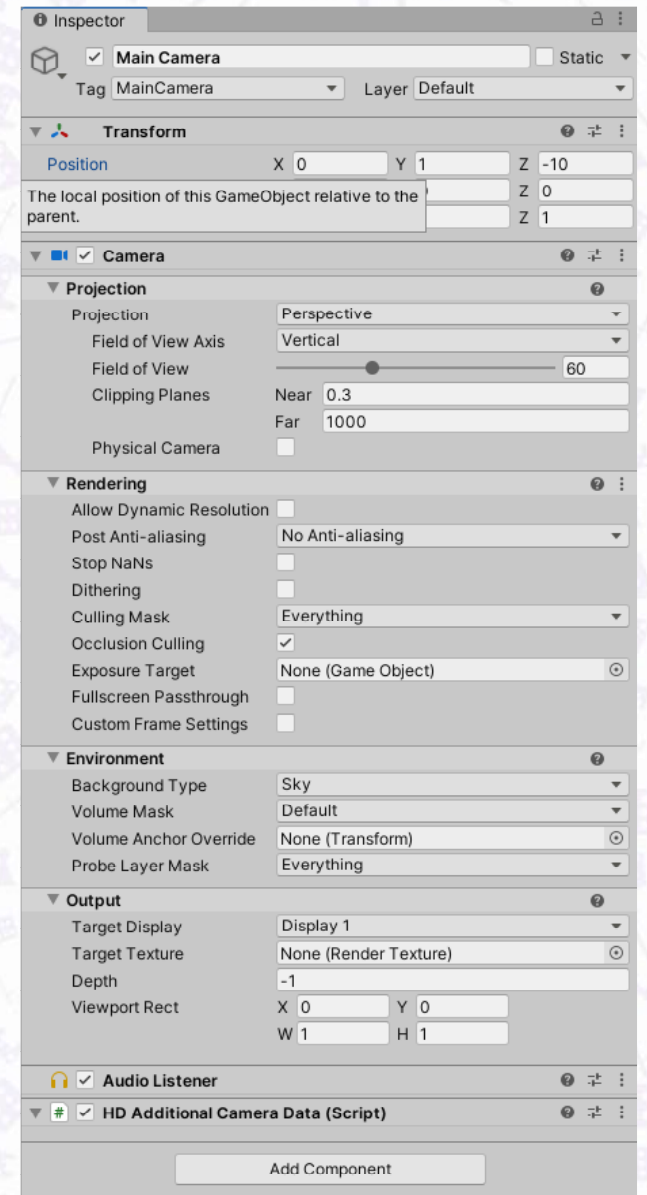


# PROGRAMOWANIE GIER

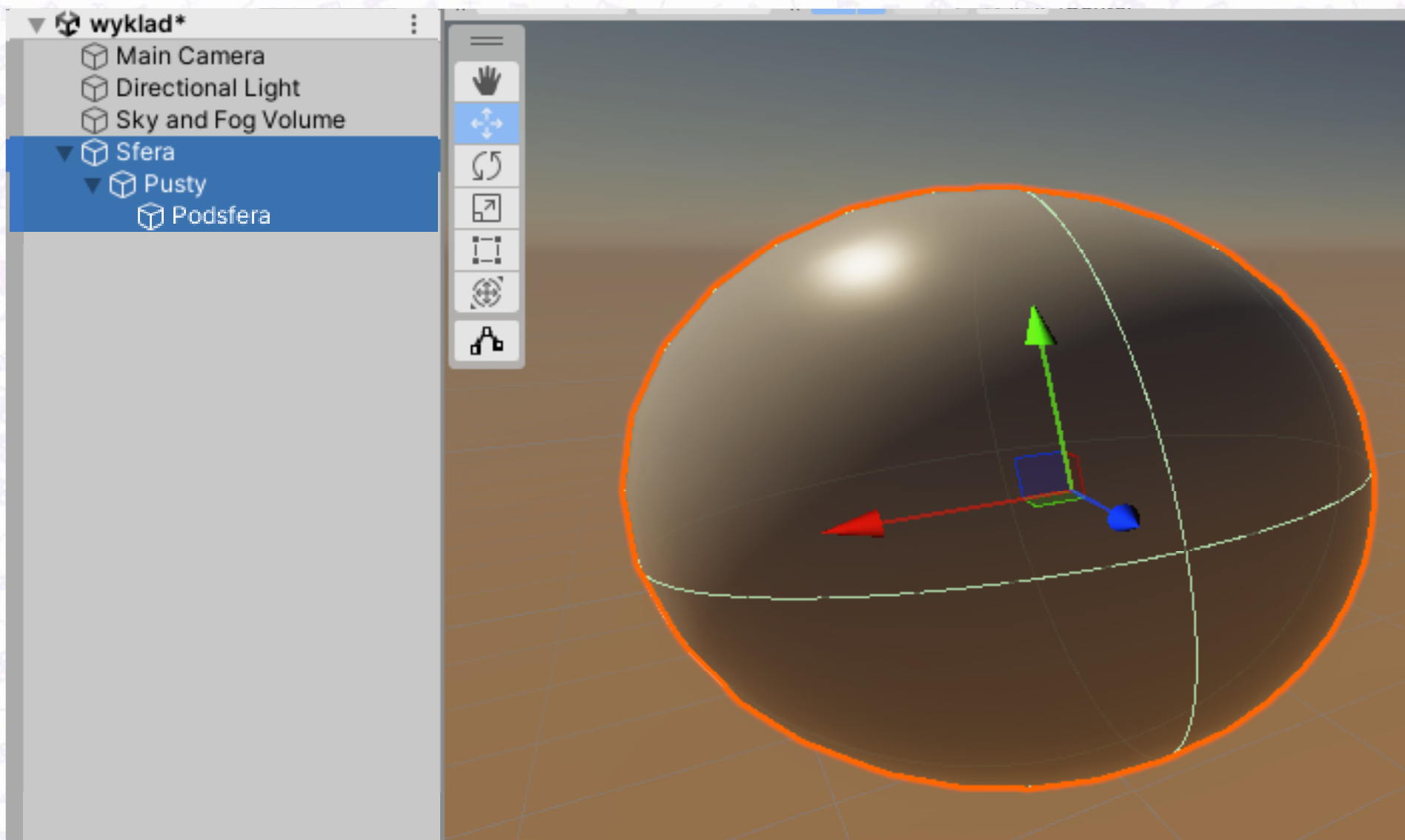
MODEL KOMPONENTOWY I TWORZENIE SKRYPTÓW

# PRZYPOMNIENIE

- Każdy obiekt składa się z komponentów
- Komponent jest klasą, która dziedziczy po klasie GameObject zdefiniowanej w aparacie silnika Unity
- Właściwości komponentów oraz ich wartości możemy zaobserwować w inspektorze
- Każdy komponent ma ściśle określone zadanie
- Klasa GameObject posiada metody, które są wywoływane automatycznie przez system zdarzeń aparatu Unity w momencie zajścia konkretnych zdarzeń
- Komponenty mogą się komunikować z innymi komponentami, które są przypisane do obiektu, a także z komponentami w innych bytach
- Każdy obiekt ma przynajmniej jeden komponent (Transform)



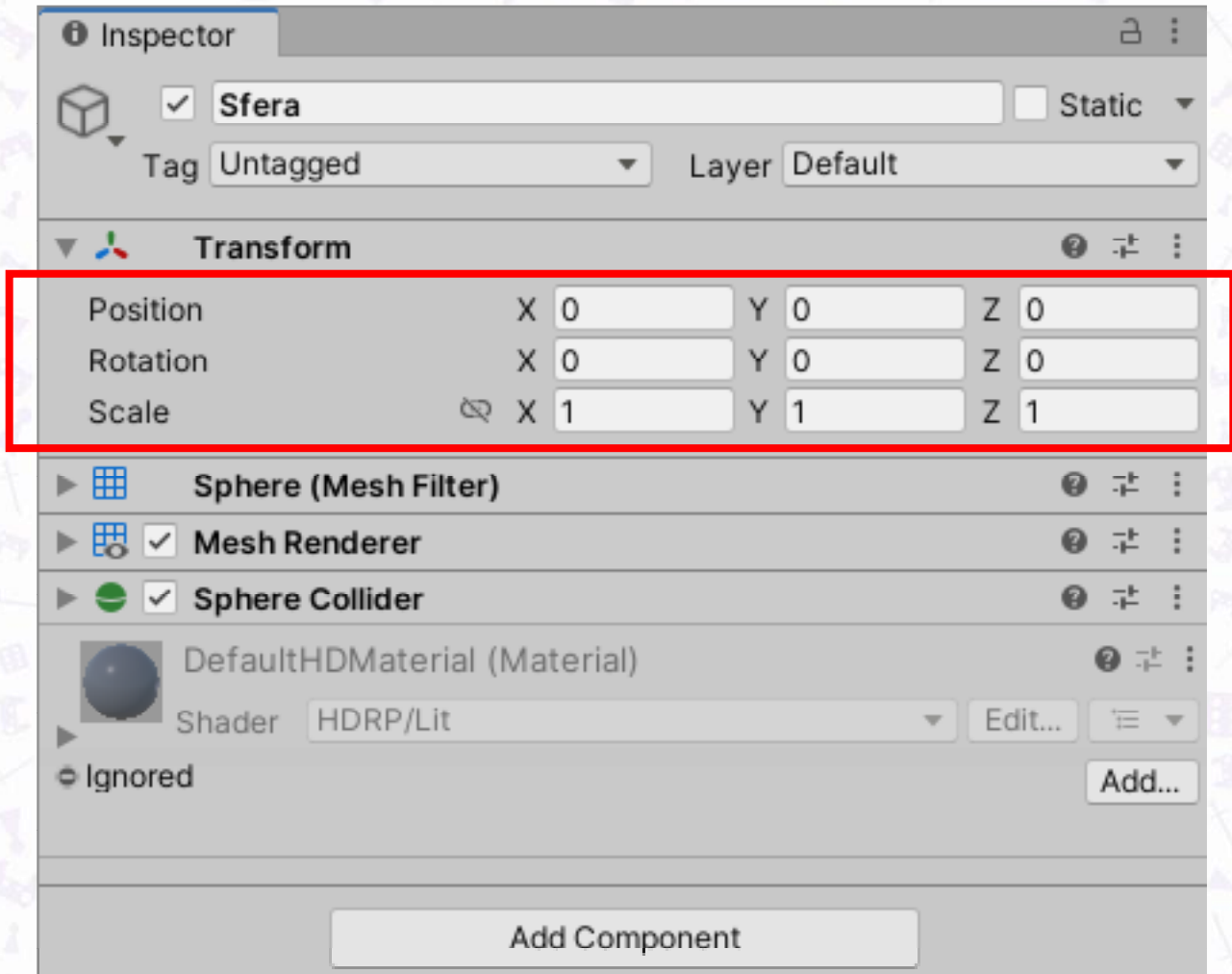
# SCENA TESTOWA



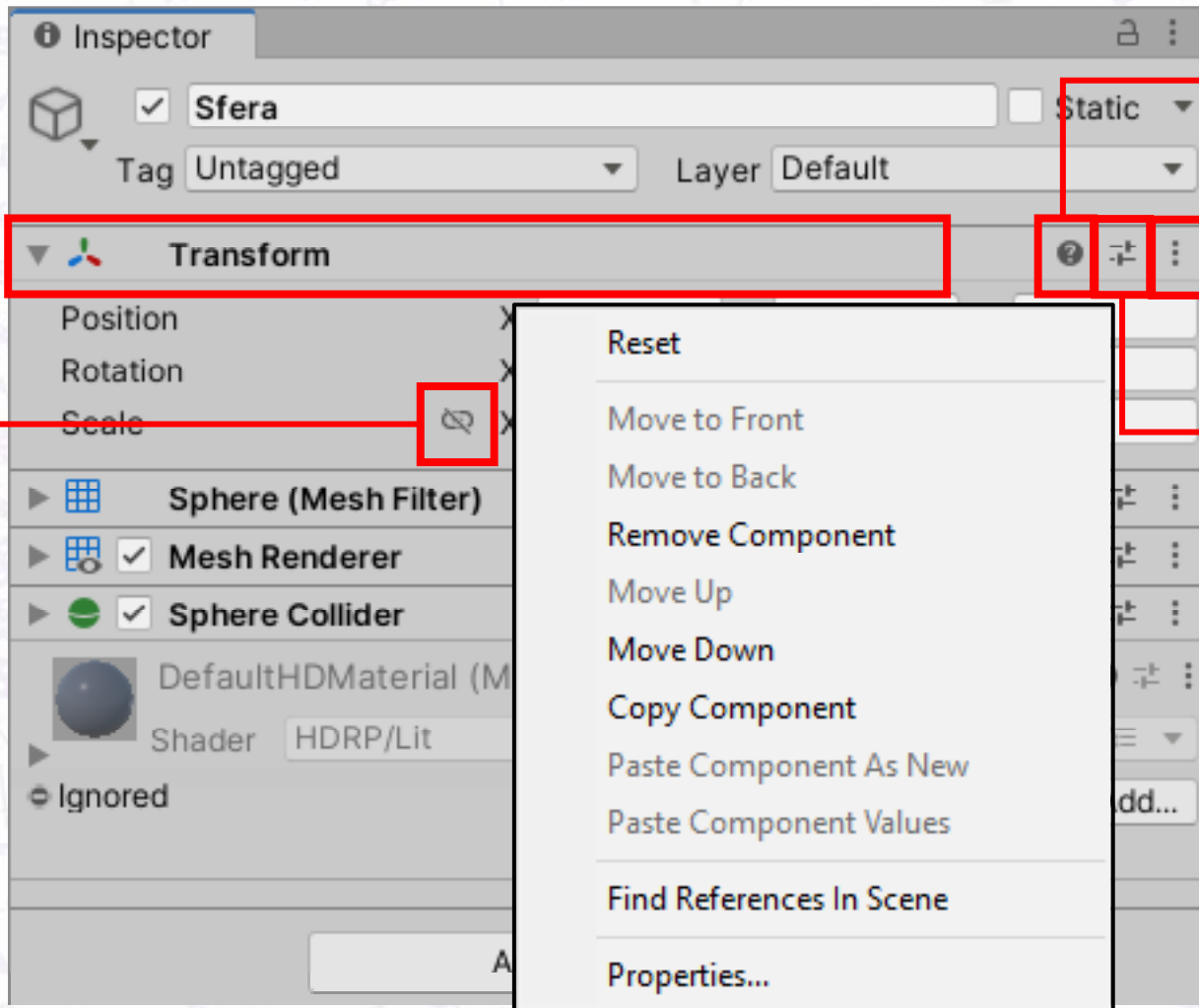
# KOMPONENTY SFERY

Byt „Sfera” ma cztery komponenty

1. Transform
2. Sphere (Mesh Filter)
3. Mesh Renderer
4. Sphere Collider



# OBSŁUGA KOMPONENTÓW



Zwijanie/rozwijanie

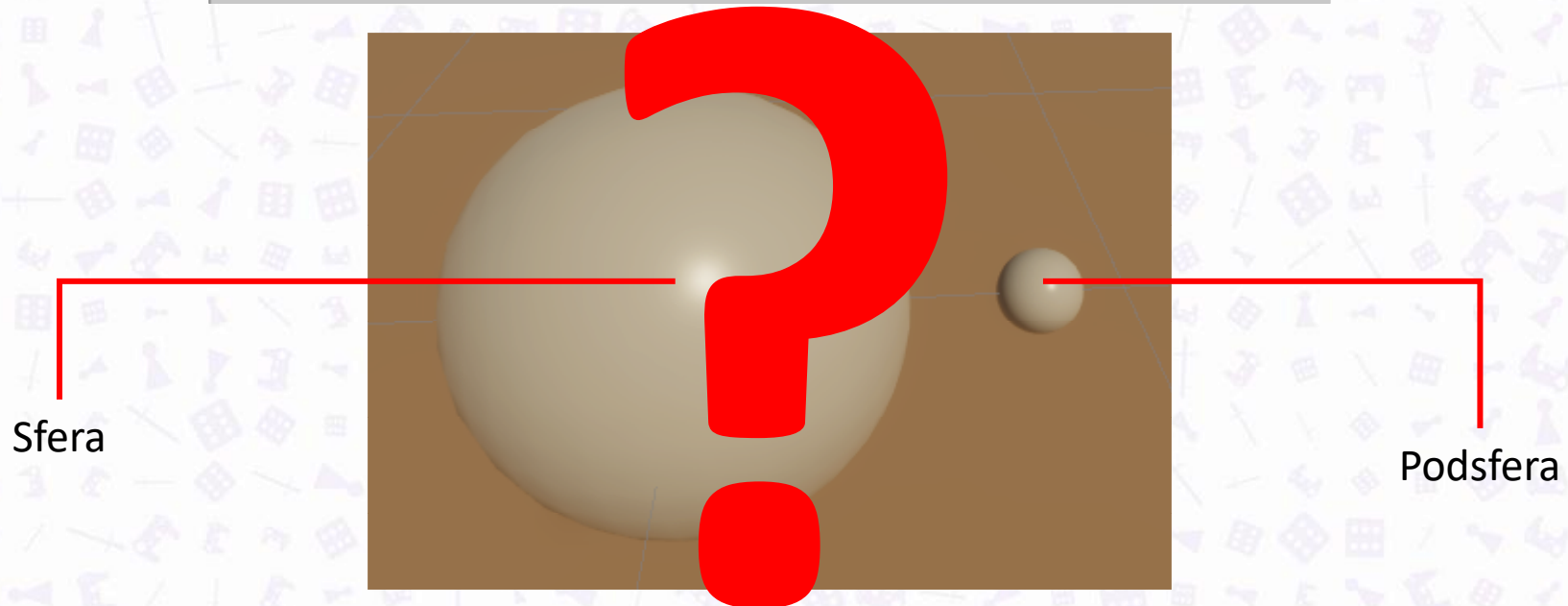
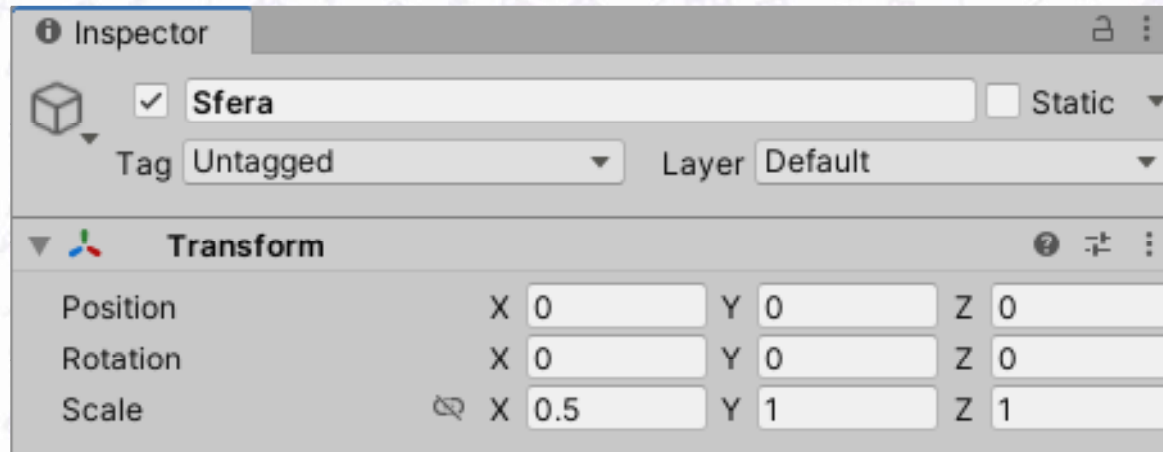
Przełączenie trybu skalowania

Przejsięcie do dokumentacji

Menu kontekstowe komponentu

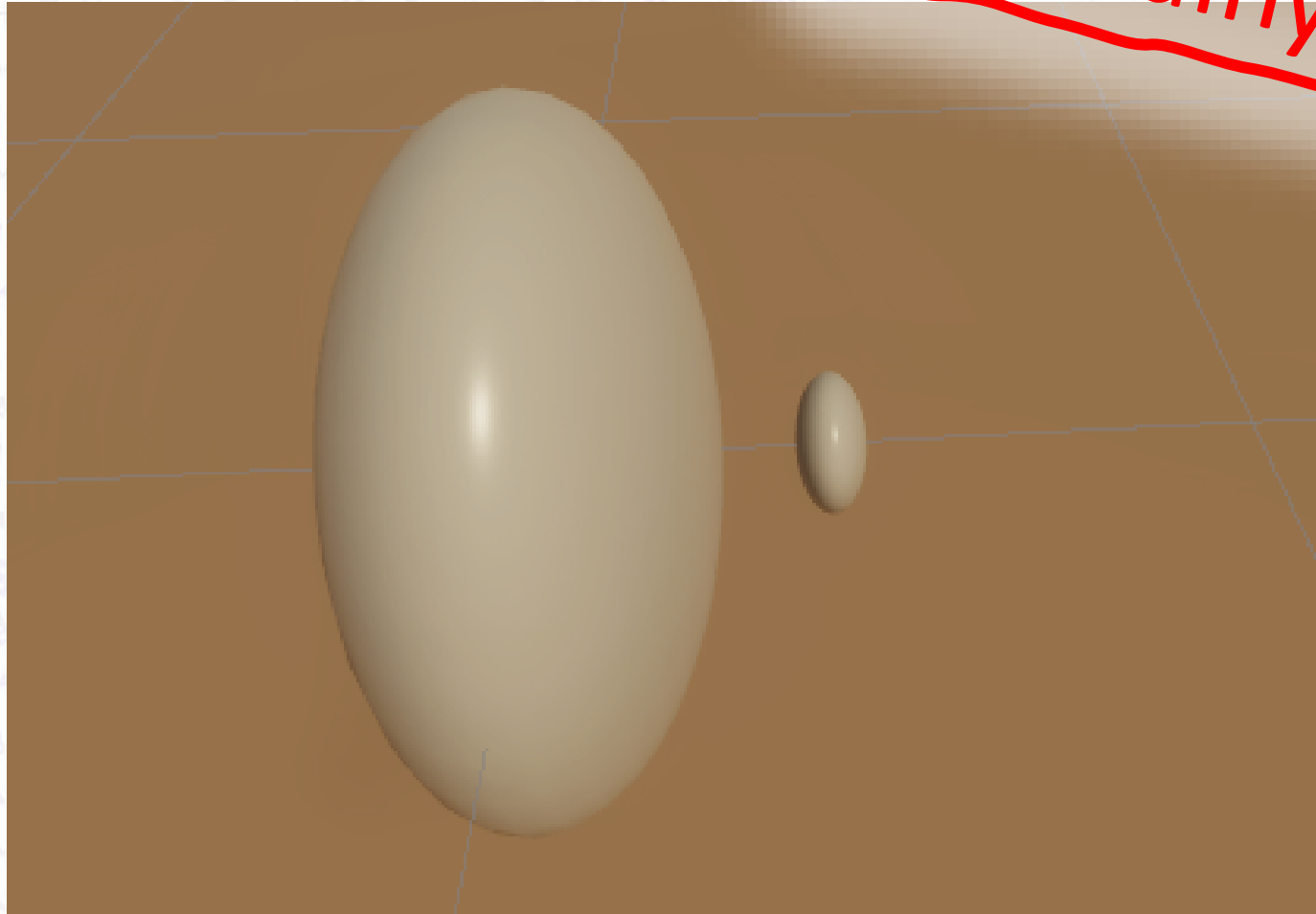
Zapis lub załadowanie wartości

# TRANSFORM: UKŁAD WSPÓŁRZĘDNYCH?



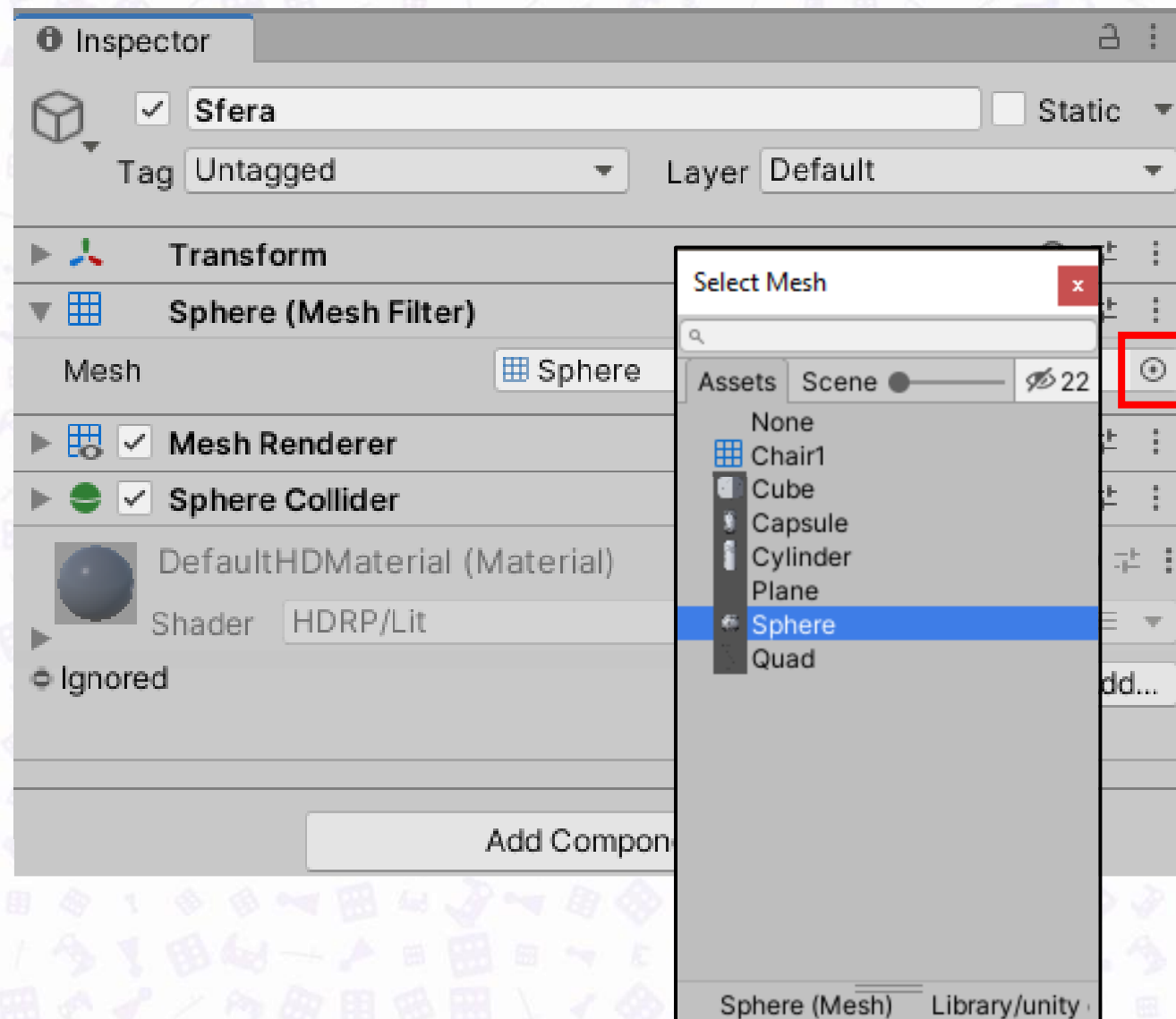
# TRANSFORM: UKŁAD WSPÓŁRZĘDNYCH?

Lokalny



# MESH FILTER

Komponent Mesh Filter przechowuje dane o punktach, krawędziach i wielokątach, z których składa się siatka modelu.

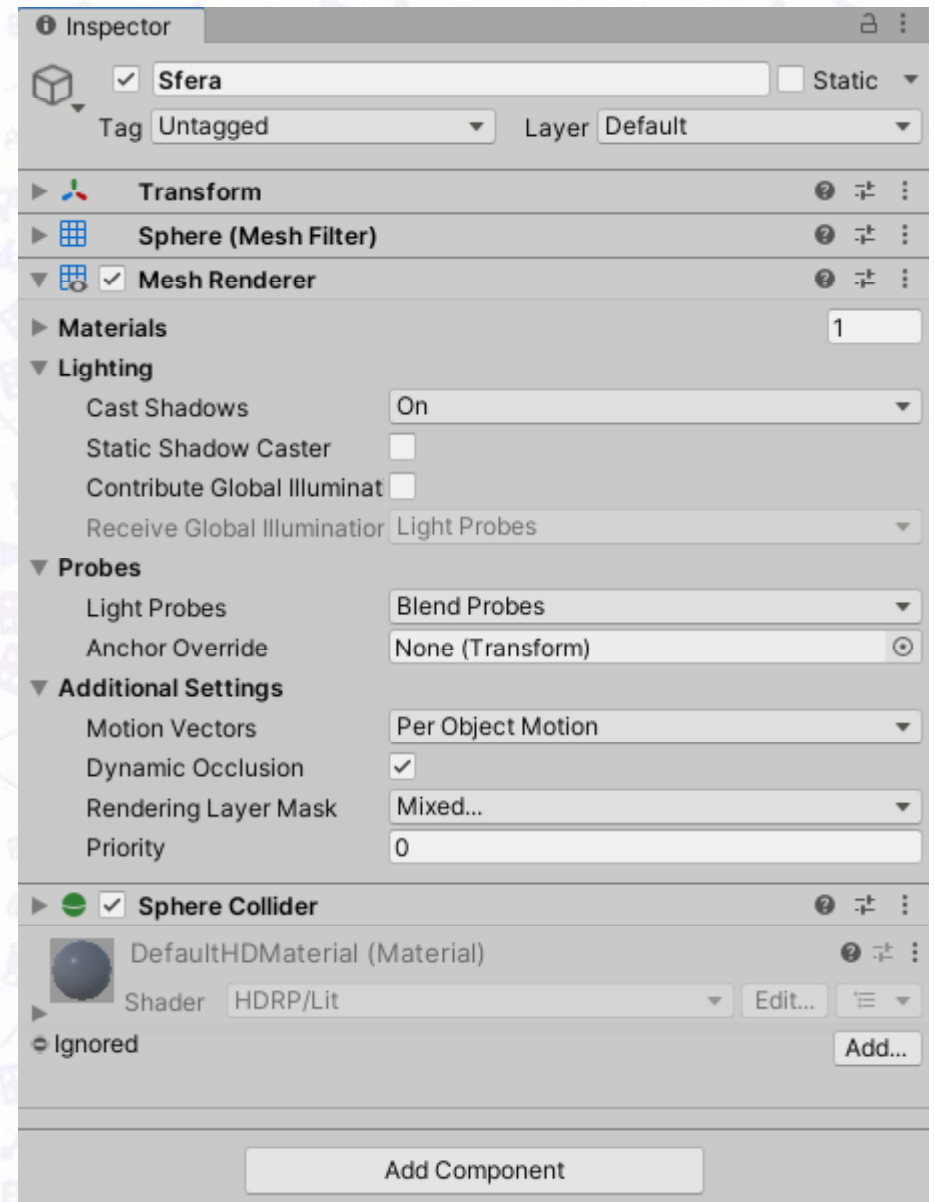




# MESH RENDERER

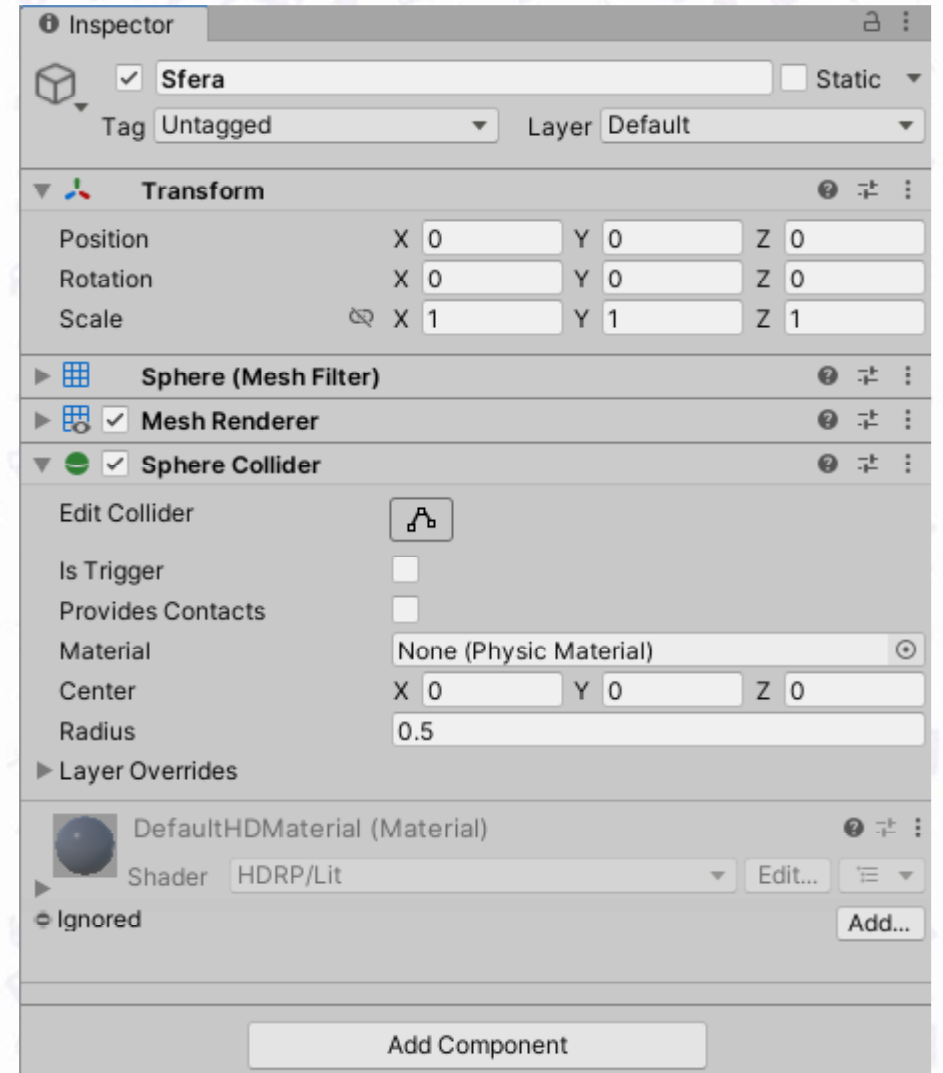
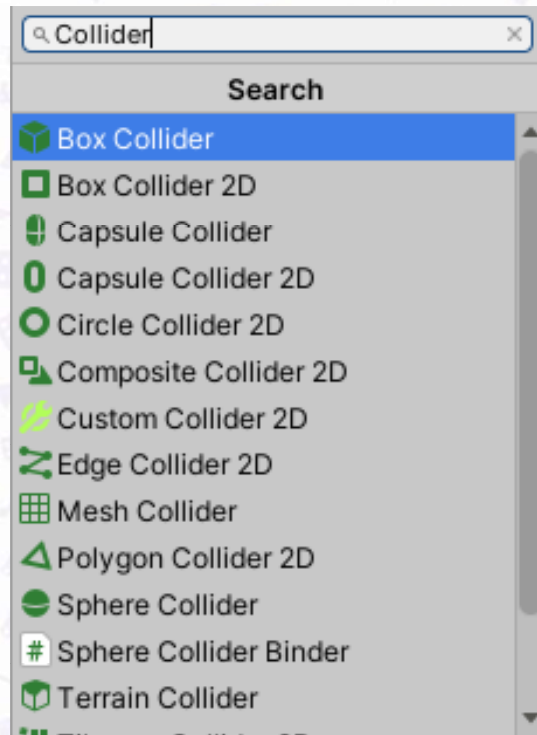
Komponent Mesh Renderer odpowiada za wyświetlanie siatki bytu, do którego jest przypisany. Pozwala również skonfigurować sposób wyświetlania oraz najważniejsze parametry:

- materiałów
- oświetlenia
- próbników
- pozostałych właściwości

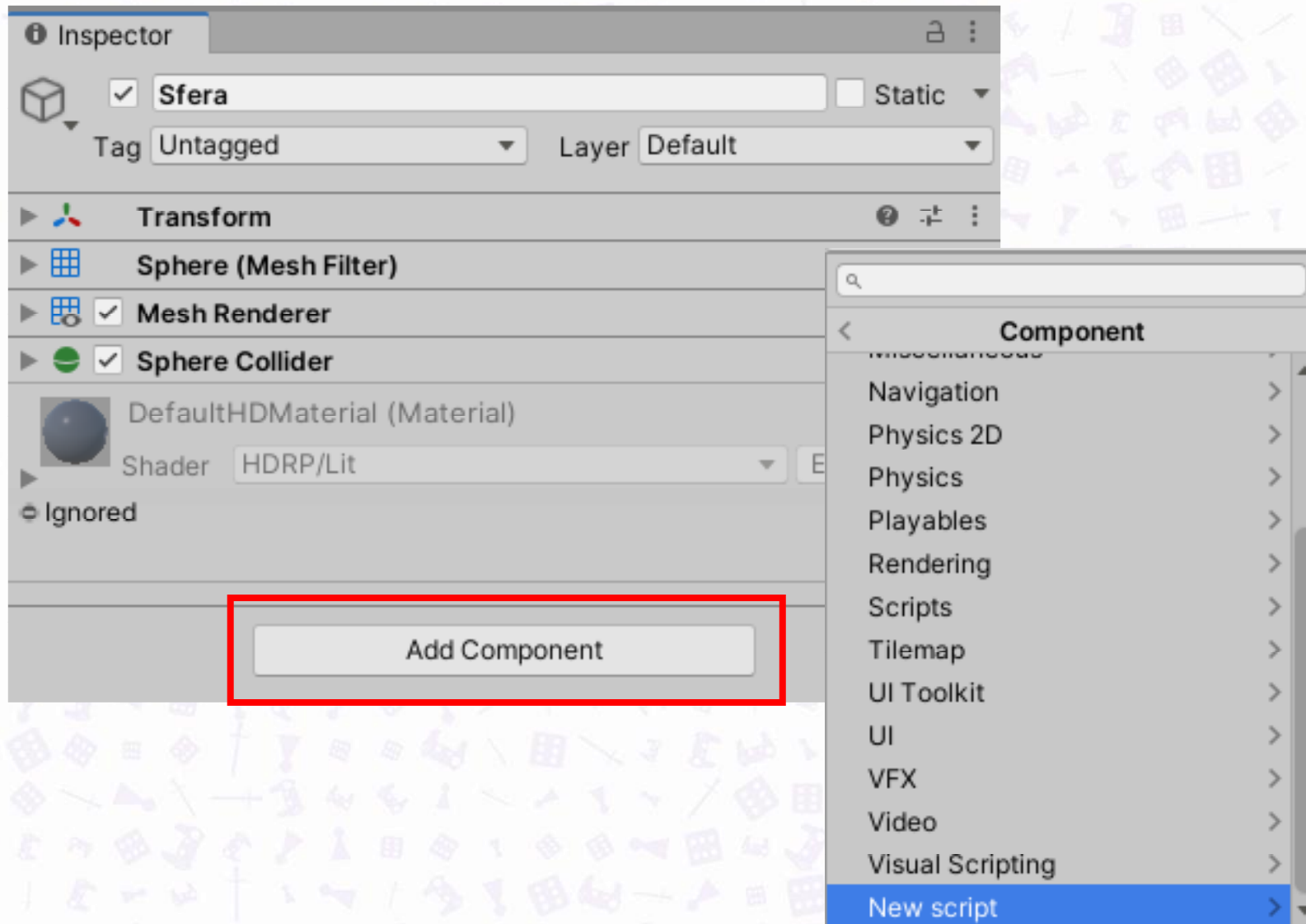


# SPHERE COLLIDER

Komponent Sphere Collider definiuje bryłę brzegową bytu („hitbox”). Jest jednym z wielu komponentów, które mają taki cel i które są zoptymalizowane pod konkretny kształt.



# DODAWANIE KOMPONENTÓW



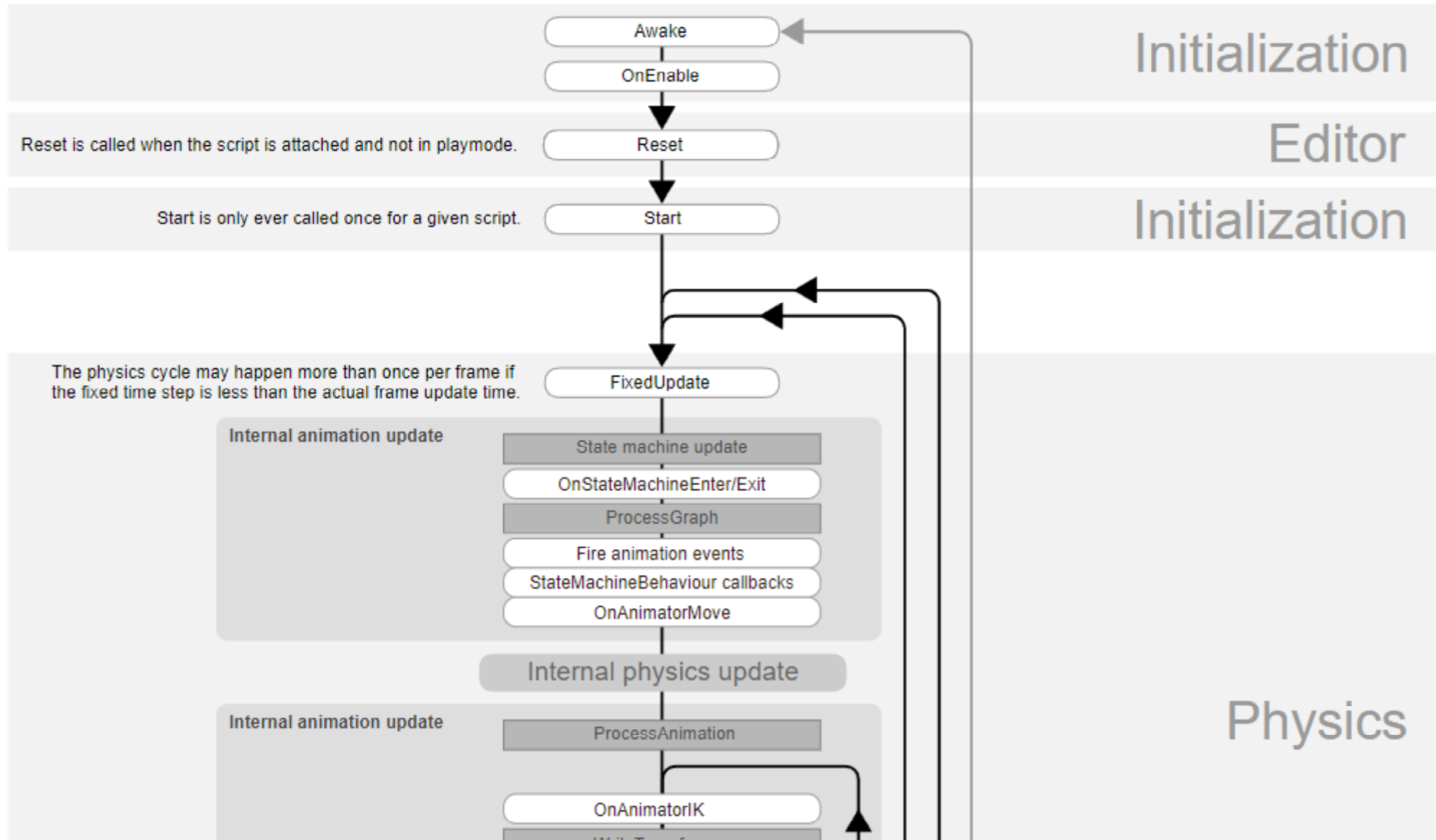
# PIERWSZY SKRYPT

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Orbitowanie : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

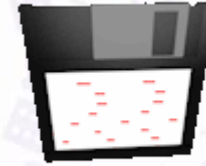
Visual Studio interface showing the code editor for 'Orbitowanie.cs'. The code is as follows:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Orbitowanie : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

# FUNKCJE APARATU UNITY

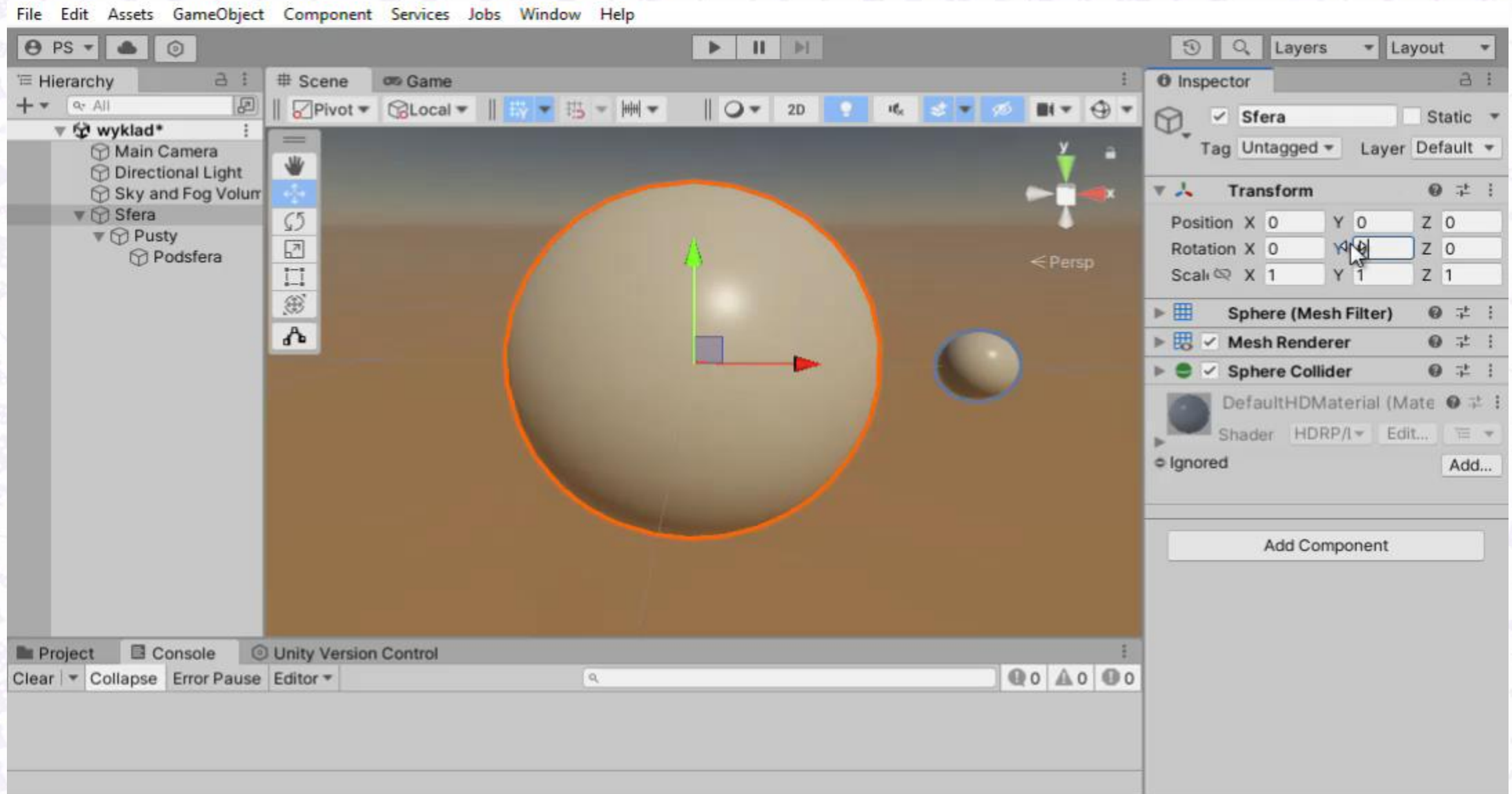


# FUNKCJE APARATU UNITY



Funkcja	Moment wywołania	Do czego wykorzystywana?
Awake	<b>Jednorazowo</b> , w momencie aktywacji bytu	Inicjalizacja, pobieranie referencji
<b>Start</b>	<b>Jednorazowo</b> , w momencie aktywacji komponentu	Inicjalizacja, nadawanie wartości początkowych
<b>Update</b>	Każdorazowo, podczas przetwarzania klatki animacji	Logika gry, interpretacja sterowania, modyfikacja sceny
FixedUpdate	Każdorazowo, w każdej klatce obliczeń fizycznych	Dodatkowe obliczenia fizyczne, wczytywanie sterowania
OnEnable	Każdorazowo, gdy komponent staje się aktywny	Tworzenie efektów cząsteczkowych, komunikacja z systemami zdarzeń
OnDisable	Każdorazowo, gdy komponent staje się nieaktywny	Tworzenie efektów cząsteczkowych, komunikacja z systemami zdarzeń

# ORBITOWANIE: USE CASE



# ORBITOWANIE: „ALGORYTM”

1. Pobierz referencję do komponentu Transform
2. W każdej klatce animacji, powtarzaj:
3. Pobierz aktualną wartość rotacji
4. Zwiększ rotację o pewną wartość
5. Wróć do kroku 2

?

**Update!**

Odczytać pole klasy...?

Po prostu dodać...?

**Update!**



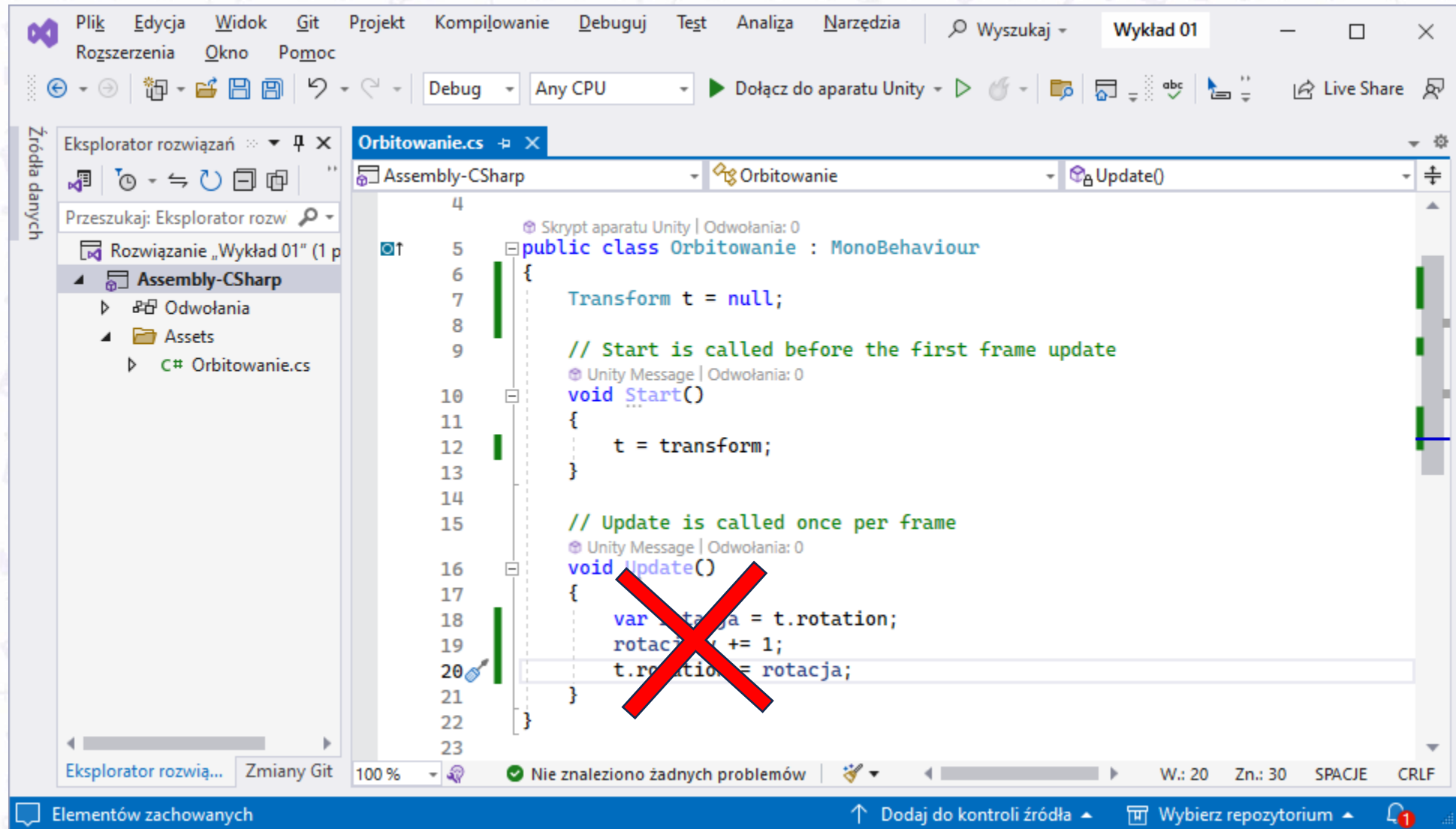
# ORBITOWANIE: KROK 1

The screenshot displays the Visual Studio IDE with the following components:

- Menu Bar:** Plik, Edycja, Widok, Git, Projekt, Kompilowanie, Debuguj, Test, Analiza, Narzędzia, Wyszukaj, Wykład 01.
- Toolbar:** Debug, Any CPU, Dołącz do aparatu Unity, Live Share.
- Left Panel (Eksplorator rozwiązań):** Shows the project structure for "Wykład 01", including "Assembly-CSharp", "Assets", and "Orbitowanie.cs".
- Code Editor:** Displays the code for "Orbitowanie.cs". The code is as follows:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 Skrypt aparatu Unity | Odwołania: 0
6 public class Orbitowanie : MonoBehaviour
7 {
8     Transform t = null;
9
10 // Start is called before the first frame update
11 Unity Message | Odwołania: 0
12 void Start()
13 {
14     t = transform;
15 }
16
17 // Update is called once per frame
18 Unity Message | Odwołania: 0
19 void Update()
20 {
21 }
22 }
```
- Status Bar:** Nie znaleziono żadnych problemów, W.: 18, Zn.: 9, SPACJE, CRLF.

# ORBITOWANIE: KROK 3 I 4



```
4
5 public class Orbitowanie : MonoBehaviour
6 {
7     Transform t = null;
8
9     // Start is called before the first frame update
10    void Start()
11    {
12        t = transform;
13    }
14
15    // Update is called once per frame
16    void Update()
17    {
18        var rotacja = t.rotation;
19        rotacja += 1;
20        t.rotation = rotacja;
21    }
22 }
23
```

Eksploator rozwiązań

Przeszukaj: Eksploator rozw

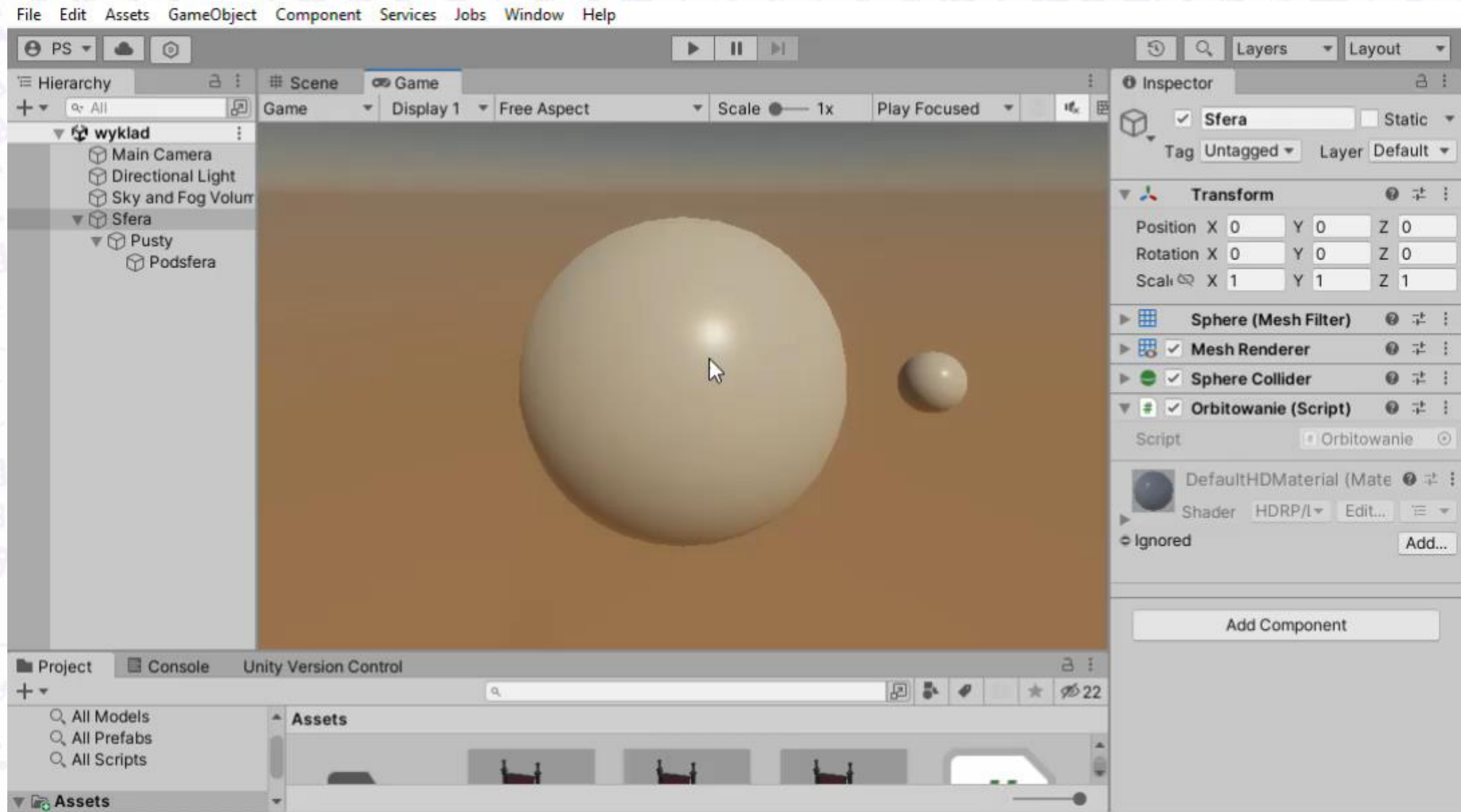
Rozwiązanie „Wykład 01” (1 p

- Assembly-CSharp
  - Odwołania
  - Assets
    - Orbitowanie.cs

Eksploator rozwią... Zmiany Git 100% Nie znaleziono żadnych problemów W.: 20 Zn.: 30 SPACJE CRLF

Elementów zachowanych Dodaj do kontroli źródła Wybierz repozytorium

# ORBITOWANIE: EFEKT



# CO MÓWI DOKUMENTACJA?

## Transform.rotation

[Leave feedback](#)

SWITCH TO MANUAL

```
public Quaternion rotation;
```



rotation stores the rotation of the Transform in world space.

rotation stores a [Quaternion](#). You can use [rotation](#) to rotate a GameObject or provide a value. Do not attempt to edit/modify [rotation](#). [Transform.rotation](#) is less than 180 degrees.

[rotation](#) has no gimbal lock.

To rotate a [Transform](#), use [Transform.Rotate](#), which uses Euler Angles.

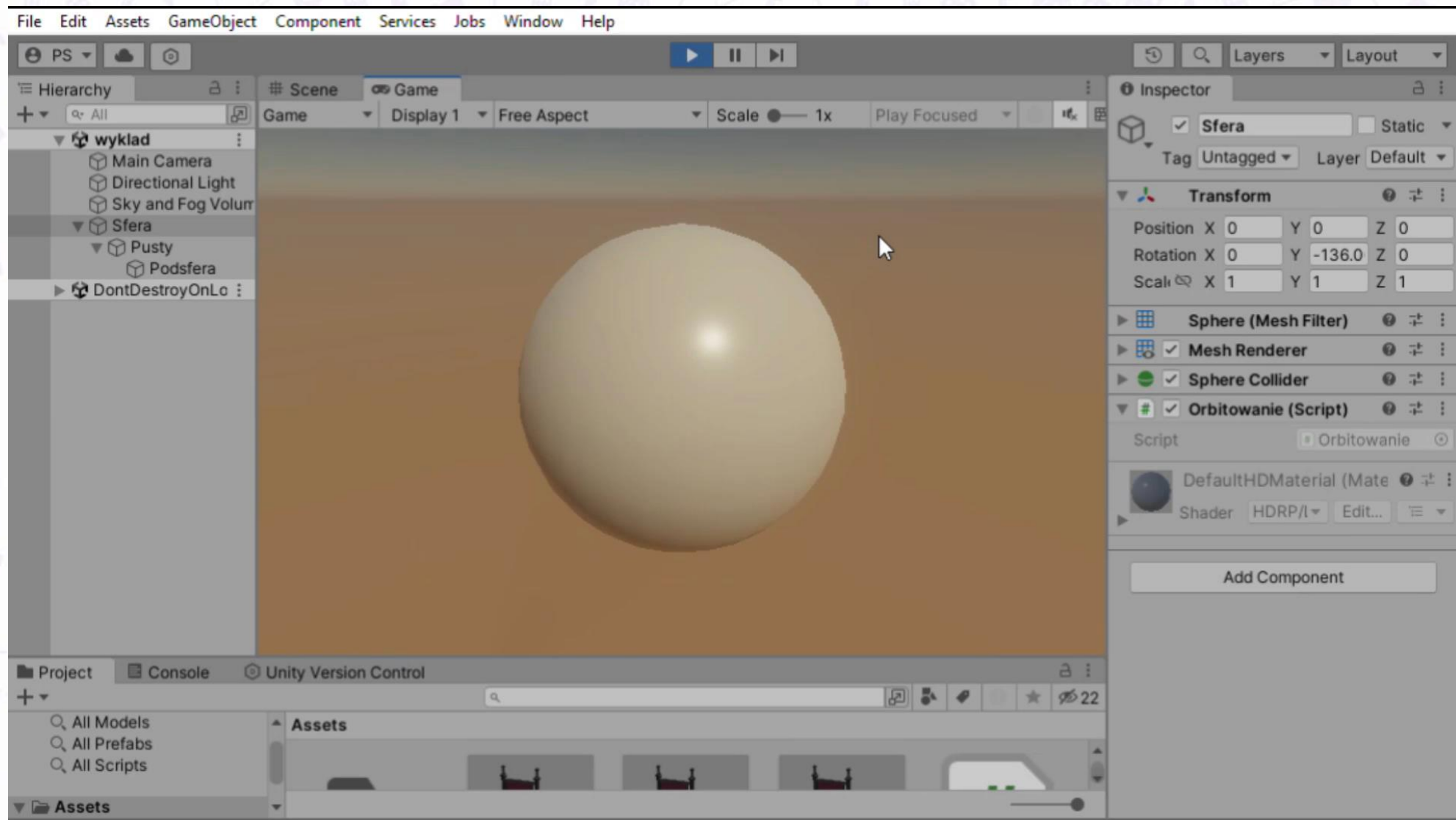
If you want to match values you see in the Inspector, use the [Quaternion.eulerAngles](#) property on the returned [Quaternion](#).



# ORBITOWANIE: KROK 3 I 4 POPRAWIONE

```
Skrypt aparatu Unity (1 odwołanie do zasobu) | Odwołania: 0
5 public class Orbitowanie : MonoBehaviour
6 {
7     Transform t = null;
8
9     // Start is called before the first frame update
10    Unity Message | Odwołania: 0
11    void Start()
12    {
13        t = transform;
14    }
15
16    // Update is called once per frame
17    Unity Message | Odwołania: 0
18    void Update()
19    {
20        var rotacja = t.eulerAngles;
21        rotacja.y += 1;
22        t.eulerAngles = rotacja;
23    }
24 }
```

# ORBITOWANIE: EFEKT



# ORBITOWANIE: ZMIENNA PRĘDKOŚĆ

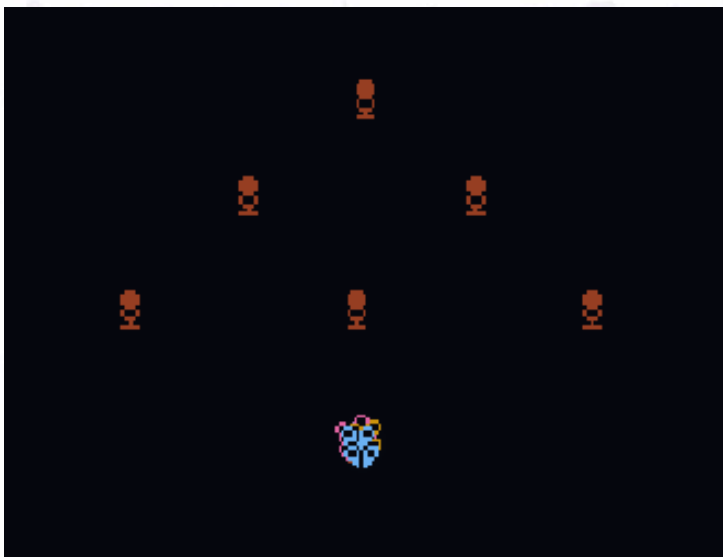
```
Skrypt aparatu Unity (1 odwołanie do zasobu) | Odwołania: 0
5 public class Orbitowanie : MonoBehaviour
6 {
7     Transform t = null;
8     public float predkosc = 1f;
9
10    // Start is called before the first frame update
11    Unity Message | Odwołania: 0
12    void Start()
13    {
14        t = transform;
15    }
16
17    // Update is called once per frame
18    Unity Message | Odwołania: 0
19    void Update()
20    {
21        var rotacja = t.eulerAngles;
22        rotacja.y += predkosc;
23        t.eulerAngles = rotacja;
24    }
}
```

The Inspector window displays the hierarchy of components on the selected object. The 'Orbitowanie (Script)' component is highlighted with a red box. Below it, the 'Predkosc' property is visible with a value of 1.

Component	Property	Value
Inspector	<input checked="" type="checkbox"/> Sfera	Static
	Tag	Untagged
	Layer	Default
Transform	Position X	0
	Y	0
	Z	0
Transform	Rotation X	0
	Y	0
	Z	0
Transform	Scale X	1
	Y	1
	Z	1
Sphere (Mesh Filter)		
<input checked="" type="checkbox"/> Mesh Renderer		
<input checked="" type="checkbox"/> Sphere Collider		
<input checked="" type="checkbox"/> Orbitowanie (Script)	Script	Orbitowanie
	Predkosc	1

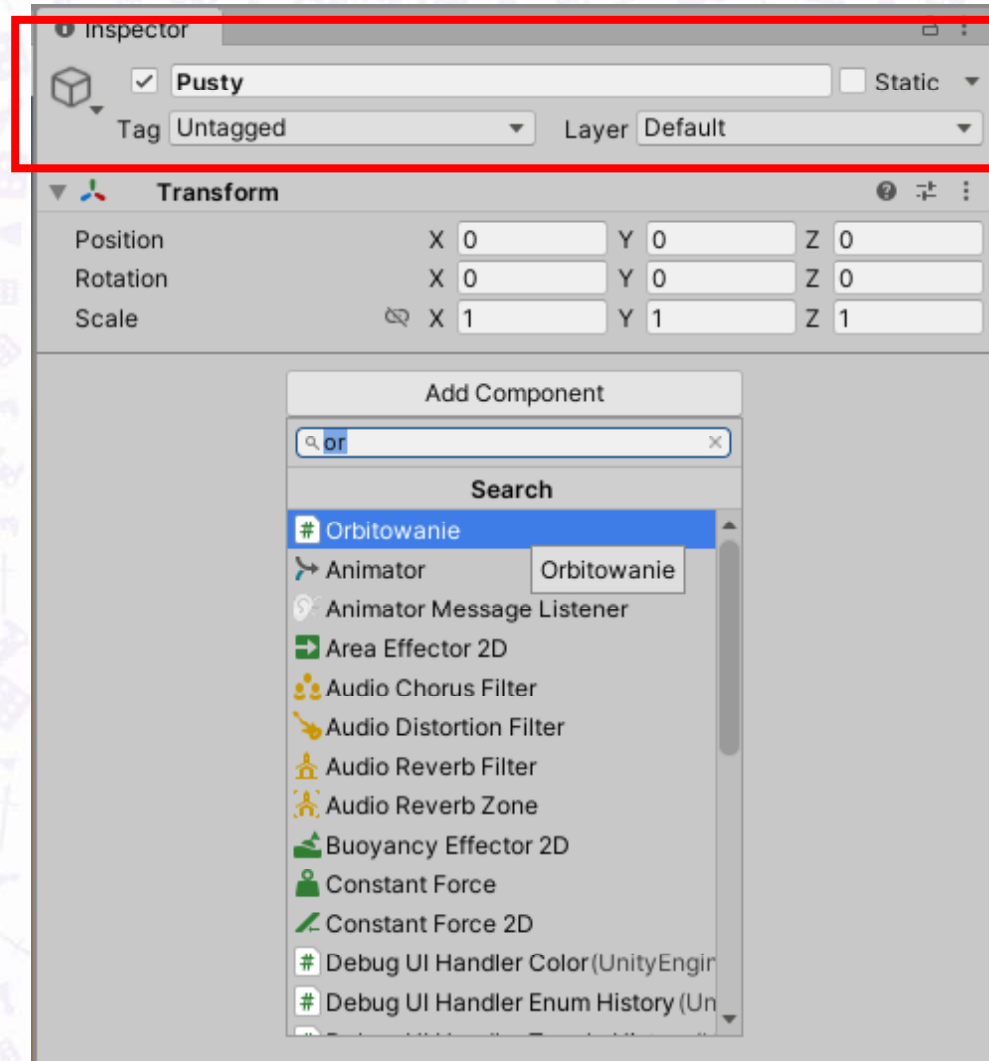
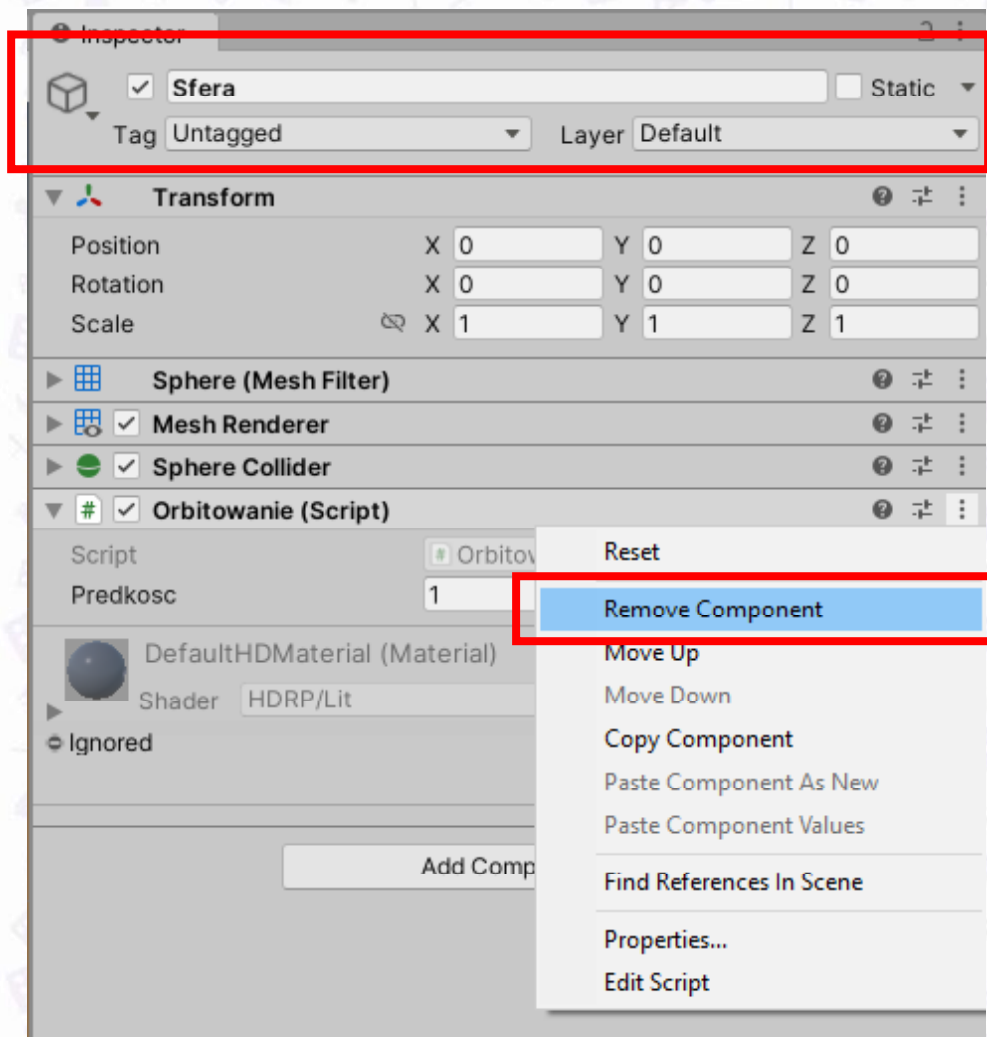
# ORBITOWANIE: PRZYDATNE?

---

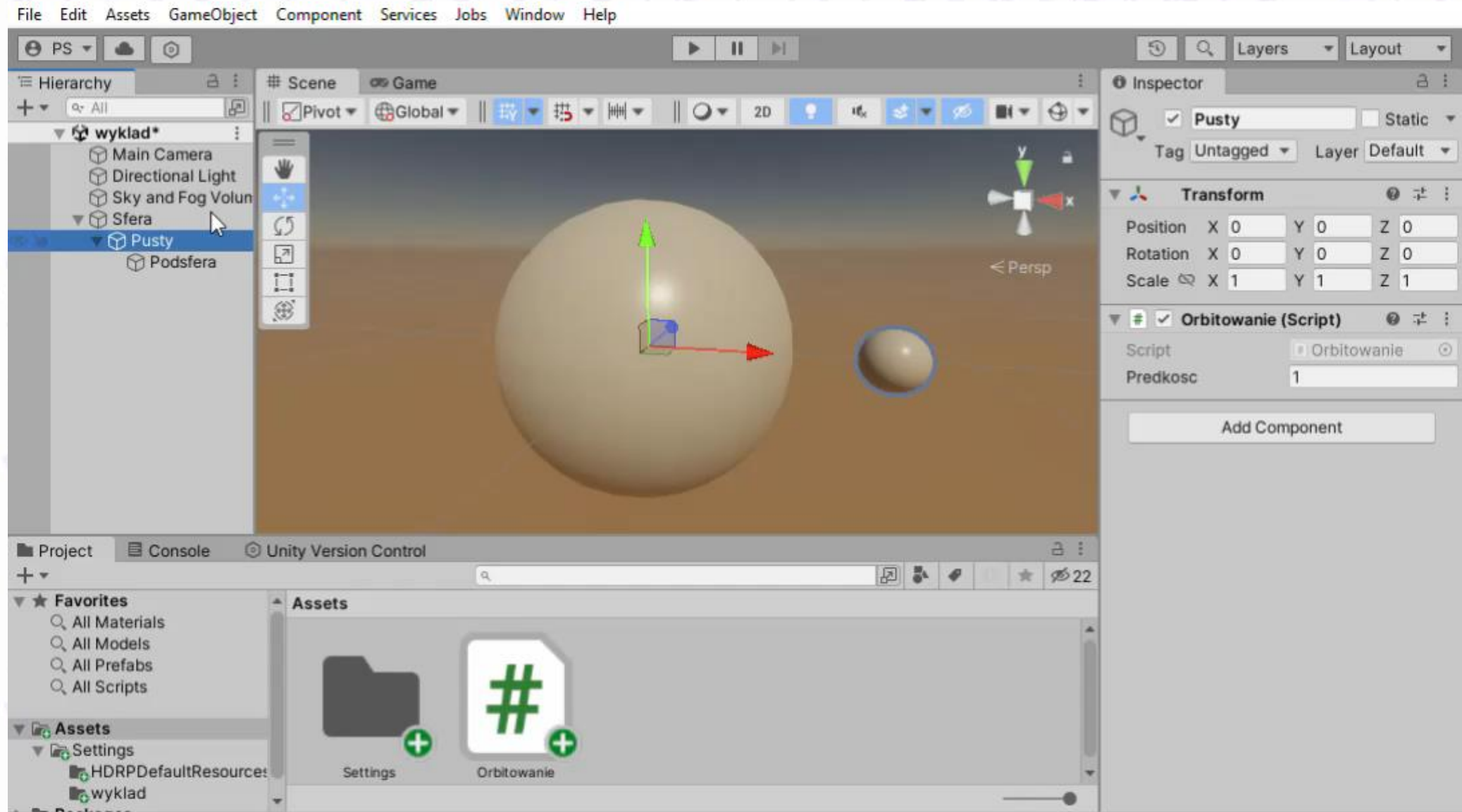




# PRZENIESIENIE SKRYPTU



# UTWORZENIE PREFABRYKATU



# INSTANTIATE

---

**Składnia:** public static Object Instantiate(Object oryginał, Transform rodzic)

**Działanie:** Tworzy kopię obiektu podanego jako pierwszy argument, ustawia jego rodzica na obiekt podany jako drugi argument, a następnie zwraca referencję do utworzonego obiektu.

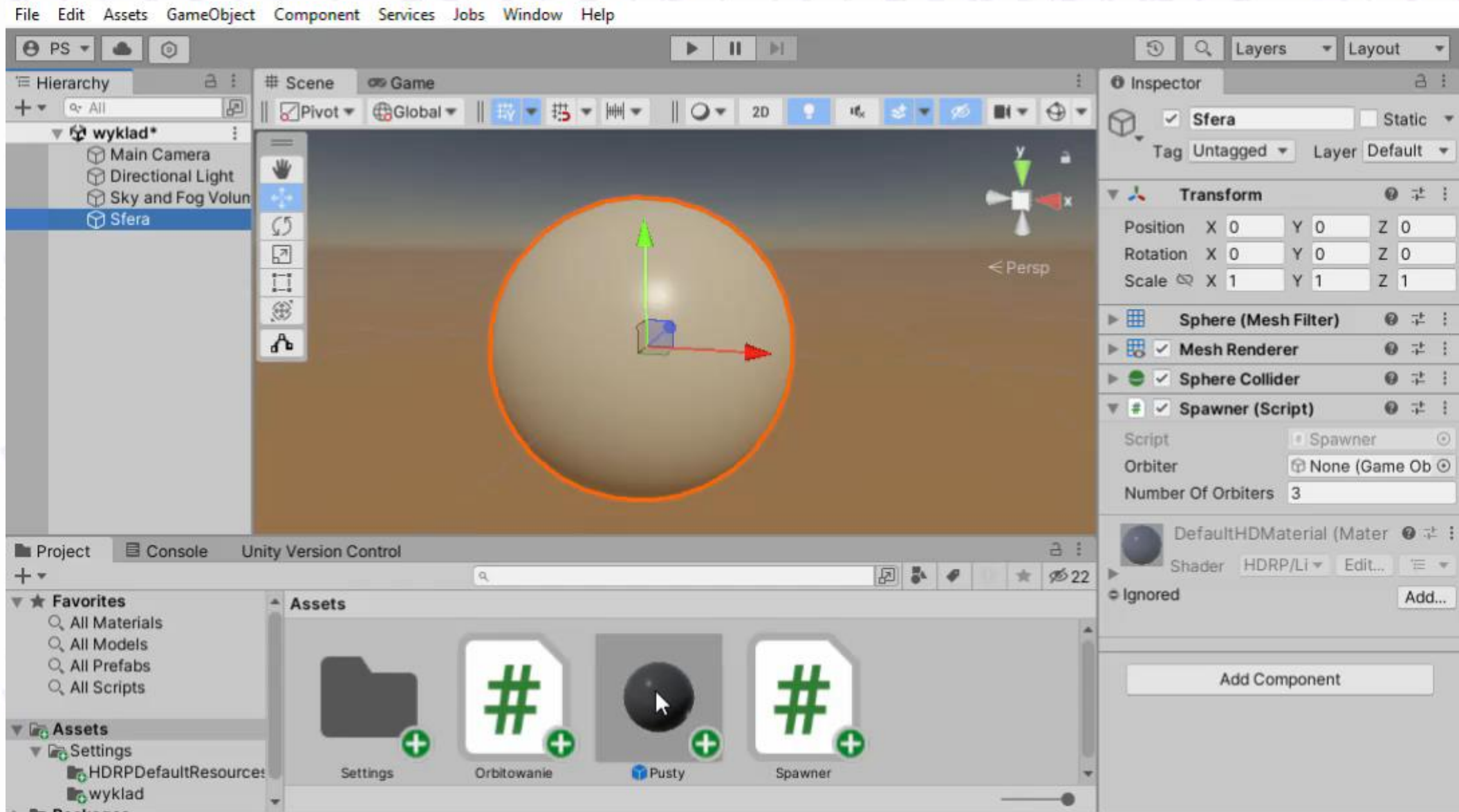
**Dokumentacja:** <https://docs.unity3d.com/ScriptReference/Object.Instantiate.html>



# SKRYPT TWORZĄCY PODSFERY

```
Skrypt aparatu Unity | Odwołania: 0
5 public class Spawner : MonoBehaviour
6 {
7     public GameObject orbiter;
8     public int numberOfOrbiters = 3;
9
10    // Start is called before the first frame update
11    Unity Message | Odwołania: 0
12    void Start()
13    {
14        for (int i = 0; i < numberOfOrbiters; i++)
15        {
16            var nowyObiekt = Instantiate(orbiter, this.transform);
17            var rotacja = nowyObiekt.transform.eulerAngles;
18            rotacja.y = 360f / numberOfOrbiters * i;
19            nowyObiekt.transform.eulerAngles = rotacja;
20        }
21    }
22 }
```

# SKRYPT TWORZĄCY PODSFERY





**DZIĘKUJĘ ZA UWAGĘ**

PROSZĘ O PYTANIA