PROGRAMOWANIE GIER

SYSTEM FIZYKI

Co mamy na myśli mówiąc „fizyka gry”?

Fizyka gry: zespół praw fizyki obowiązujących w ramach danej gry lub symulacji, także implementujący te prawa kod źródłowy, obliczone zgodnie z tymi prawami i odtwarzane animacje oraz wszystkie elementy, które te prawa imitują.

W szczególności za fizykę gry uznaje się:

- Symulację wzajemnego przyciągania się obiektów
- Obsługę kolizji
- Animacje proceduralne (np. animację szkieletową, ragdoll)
- Systemy cząsteczkowe
- Symulację cieczy
- Symulację ciał sztywnych (drewno, metal, szkło)
- Symulację ciał miękkich (włosy, futro, materiał, szkło)


I AM
BREAD

RODZAJE CIAŁ/BRYŁ

1. **Ciało sztywne** (ang. „*rigid body*”): obiekt, definiowany przez siatkę wielokątów, której punkty nie ulegają przemieszczeniu w trakcie symulacji. *Przykłady: drewniana ściana, metalowy słup, szklany wieżowiec.*
2. **Ciało miękkie** (ang. „*soft body*”): obiekt, którego siatka zawiera punkty, których wzajemna odległość może ulec zmianie w trakcie symulacji, w reakcji na bodźce fizyczne. *Przykłady: włosy, tkanina, ciecz, maź, szklane okno.*

Bodźce fizyczne

1. Siły zewnętrzne
2. Kolizje z innymi obiektami
3. Upływ czasu

Symulacja fizyki w silniku Unity

- Osobny silnik do symulacji obliczeń fizyki dwuwymiarowej i trójwymiarowej
- Możliwa jest zmiana silnika fizycznego poprzez pobranie odpowiedniego pakietu za pomocą zarządcy pakietów
- Do opisu brył i właściwości fizycznych służą komponenty, do opisu zachowań służą ich parametry i funkcje
- Obliczenia fizyczne są przeprowadzane co 20ms, 50 razy na sekundę
- Implementację dodatkowych elementów fizyki należy przeprowadzić w funkcji FixedUpdate lub funkcjach dedykowanych obsłudze konkretnych zdarzeń fizycznych

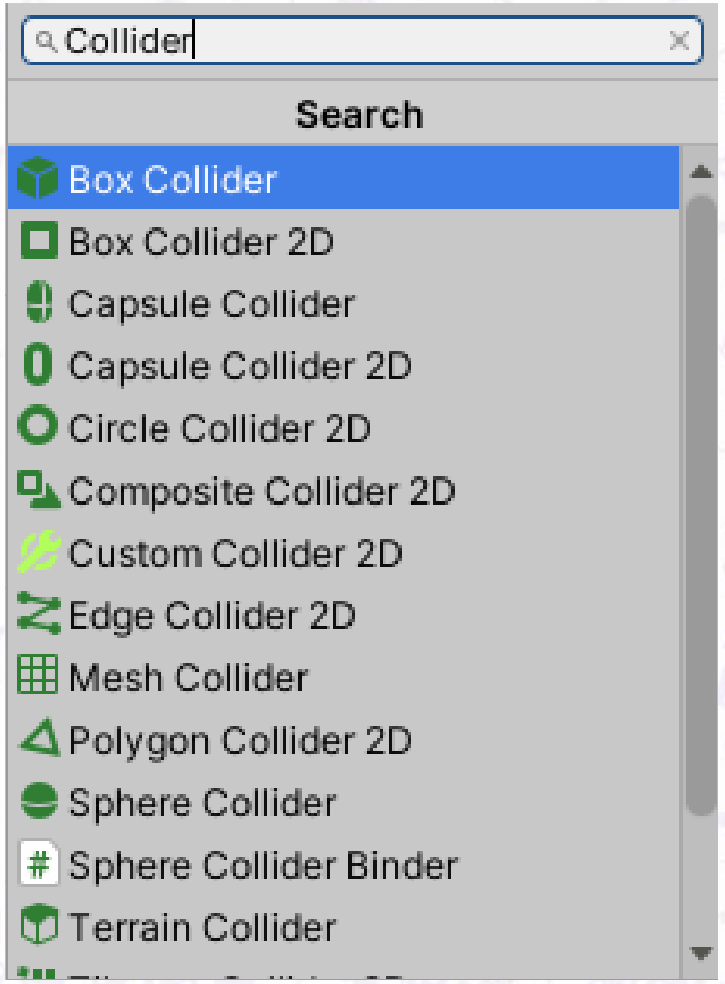
PODSTAWOWE BRYŁY BRZEGOWE

Płaskie bryły brzegowe:

- Box Collider 2D – prostokąt
- Capsule Collider 2D – zaokrąglony prostokąt
- Circle Collider 2D – kółko

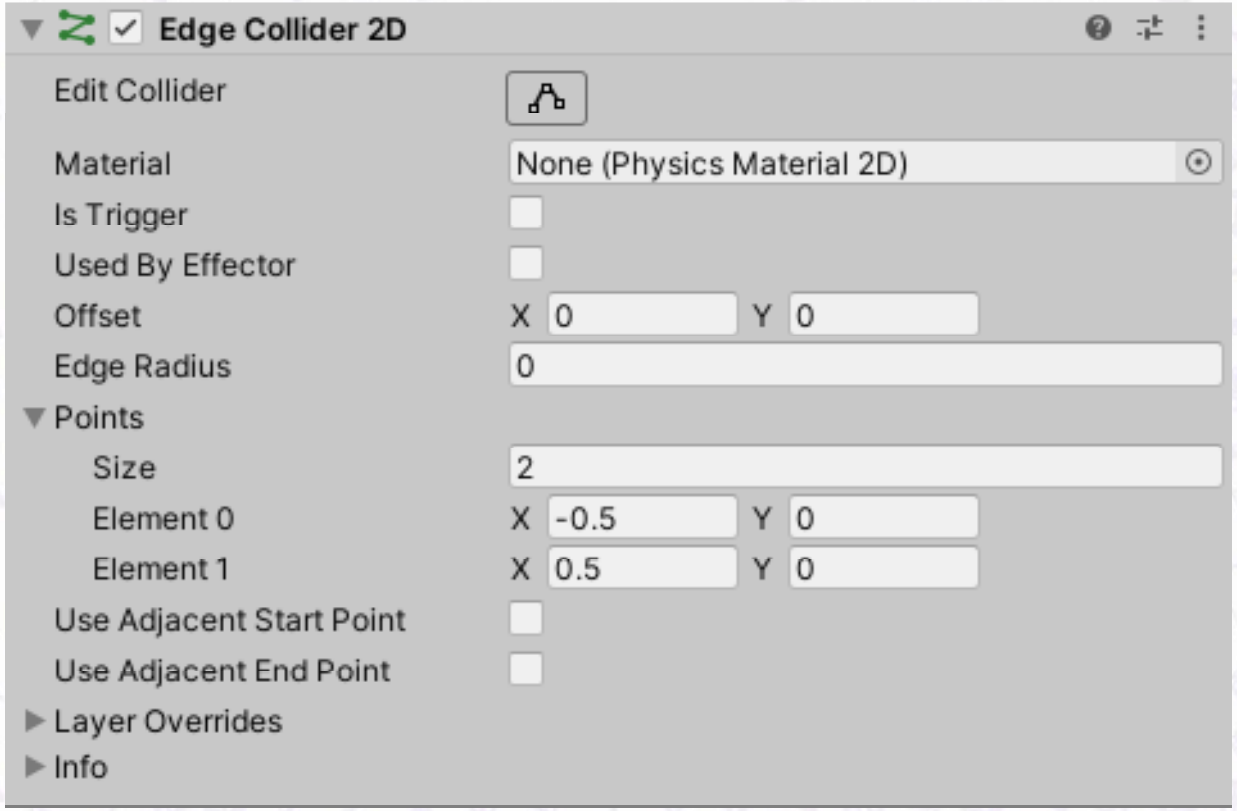
Trójwymiarowe bryły brzegowe:

- Box Collider – prostopadłościan
- Capsule Collider – „tabletka”
- Sphere Collider – kula



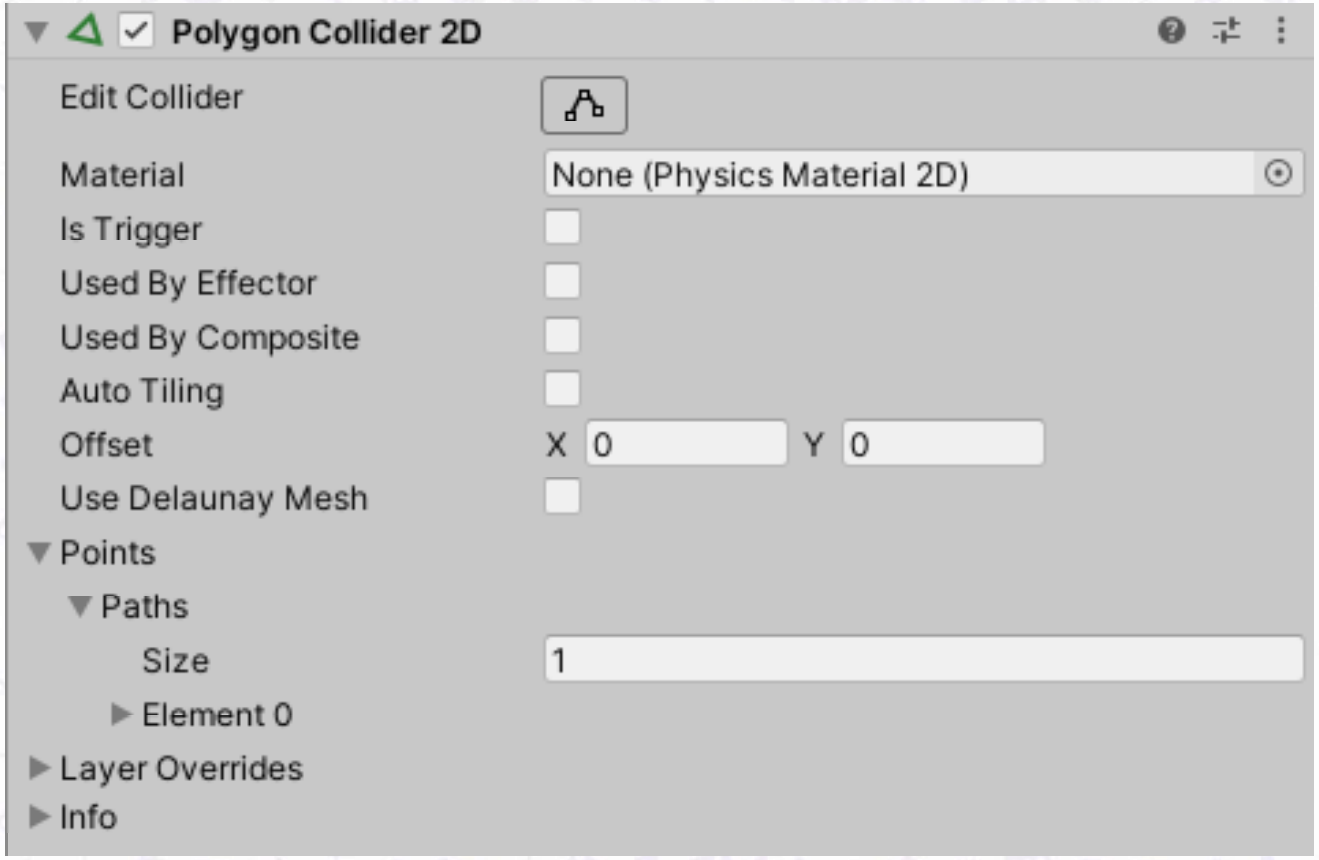
Edge Collider 2D

Komponent opisuje płaską bryłę brzegową będącą linią łamaną otwartą lub zamkniętą.



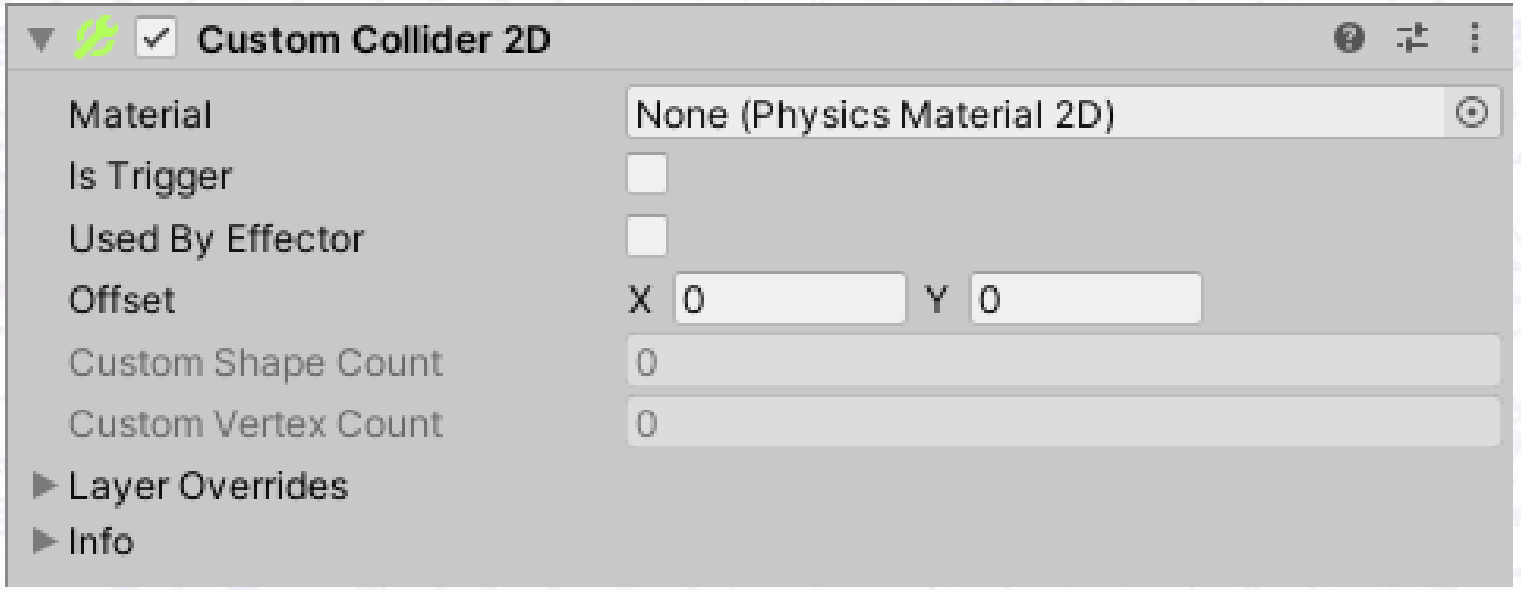
Polygon Collider 2D

Komponent opisuje płaską bryłę brzegową będącą powierzchnią ograniczoną linią łamaną zamkniętą.



Custom Collider 2D

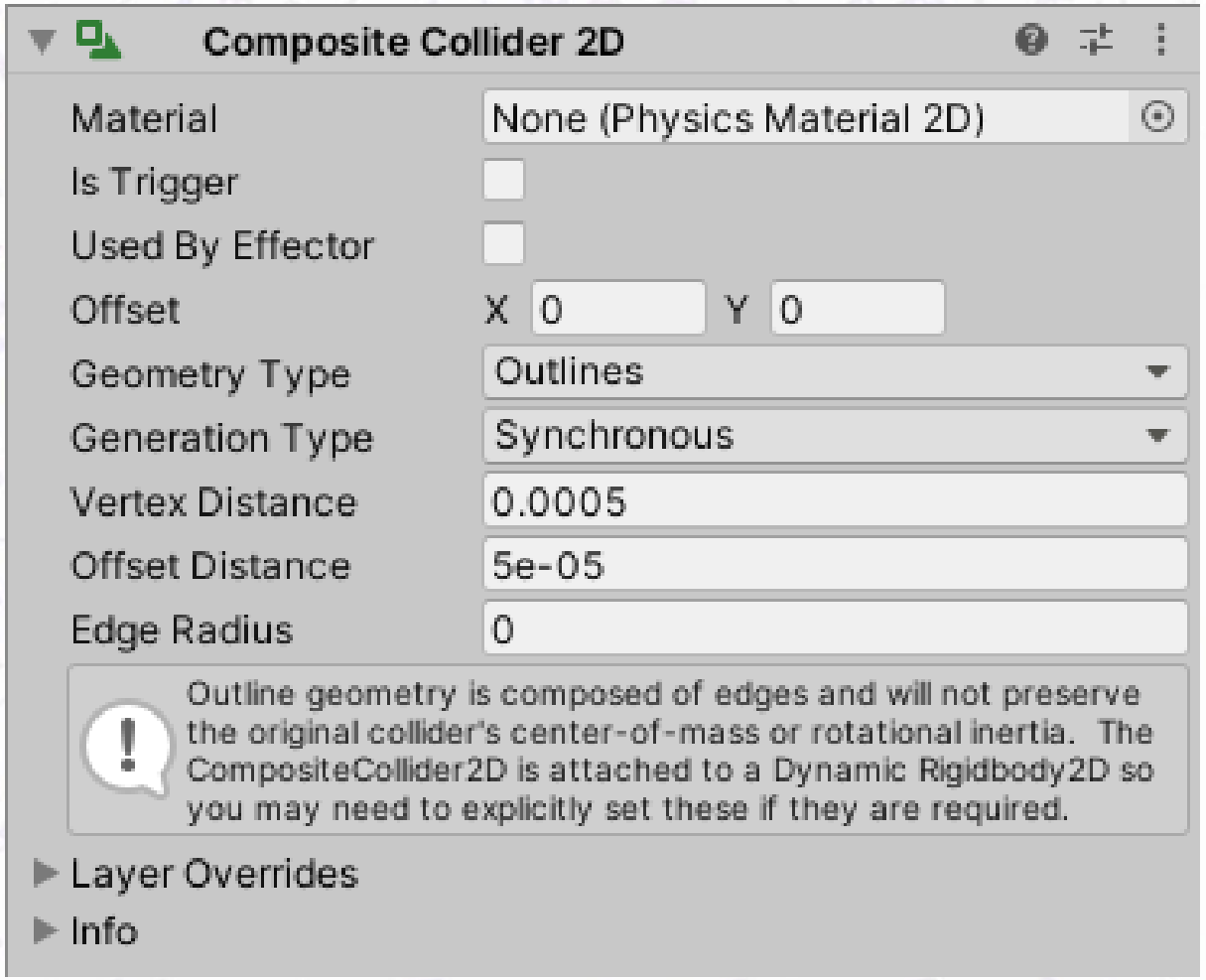
Komponent opisuje płaską bryłę brzegową będącą powierzchnią ograniczoną linią łamaną zamkniętą.



Composite Collider 2D

Komponent opisuje płaską bryłę brzegową złożoną z innych brył brzegowych definiowanych przez następujące komponenty:

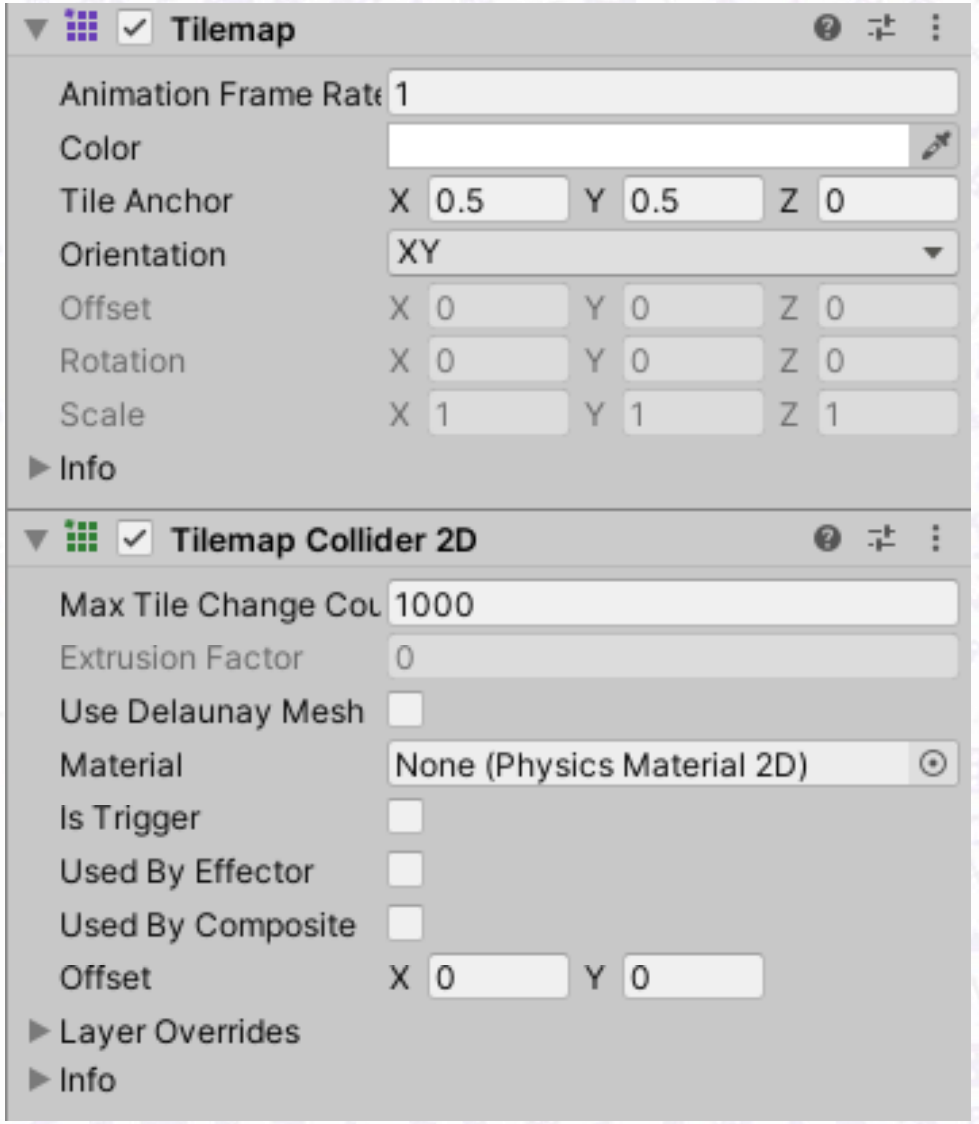
- BoxCollider2D (prostokąt)
- PolygonCollider2D (wielokąt)



TILEMAP COLLIDER 2D

Komponent opisuje płaską bryłę brzegową, której kształt jest zależny od zawartości komponentu Tilemap.

https://learn.unity.com/tutorial/introduction-to-tilemaps

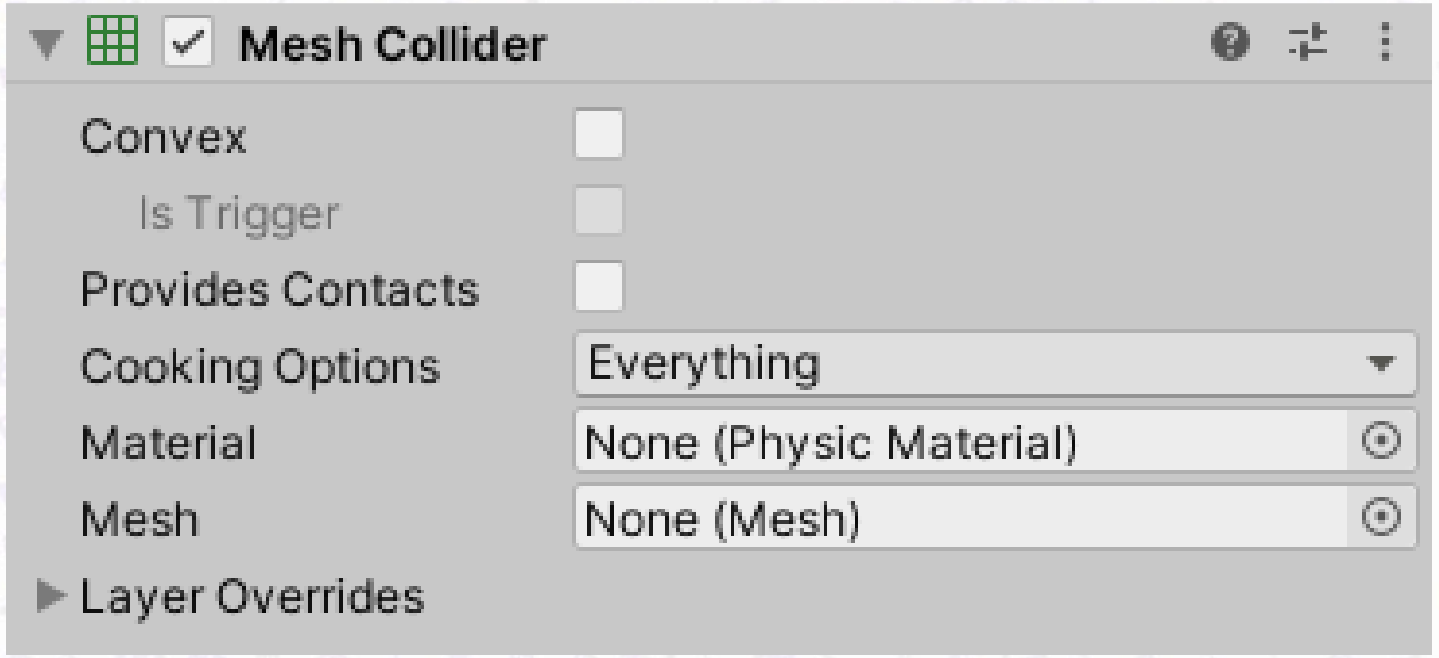


ZAAWANSOWANE PŁASKIE BRYŁY BRZEGOWE

Komponent	Kształt
EdgeCollider2D	Łamana
PolygonCollider2D	Powierzchnia ograniczona łamaną zamkniętą
CustomCollider2D	Powierzchnia zdefiniowana listą obiektów klasy PhysicsShapeGroup2D
CompositeCollider2D	Suma powierzchni lub krawędzi komponentów BoxCollider2D i PolygonCollider2D
TilemapCollider2D	Zależny od kafelków, z których utworzono dany fragment mapy

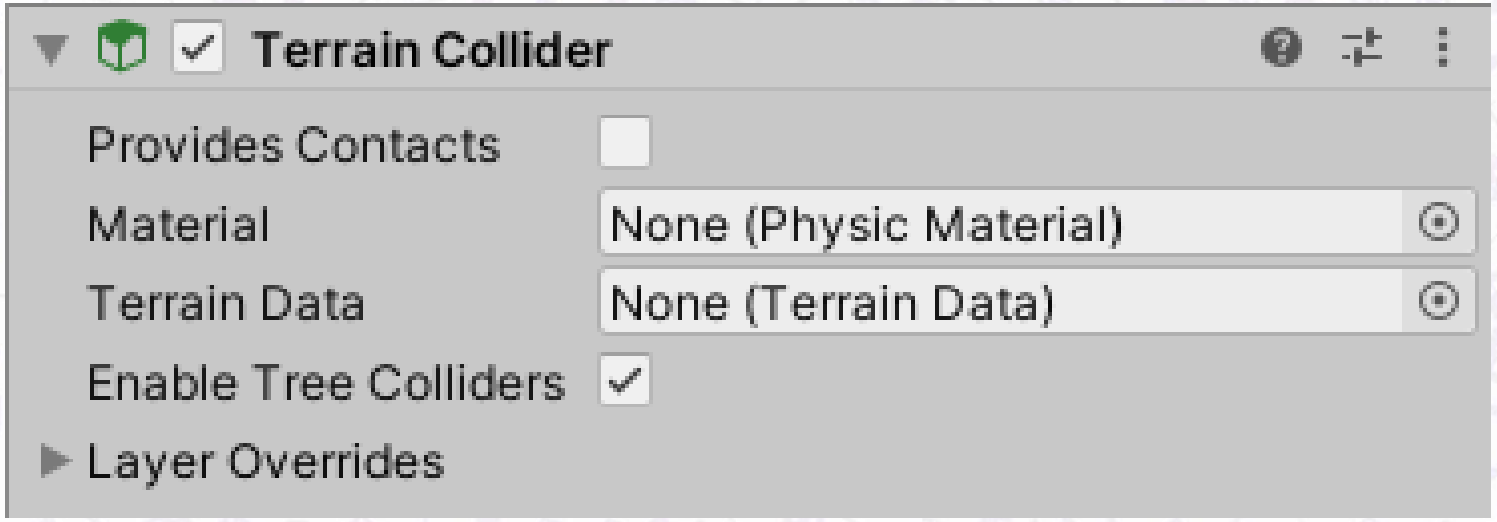
Mesh Collider

Komponent opisuje bryłę brzegową będącą siatką wielokątów



Terrain Collider

Komponent opisuje bryłę brzegową komponentu Terrain



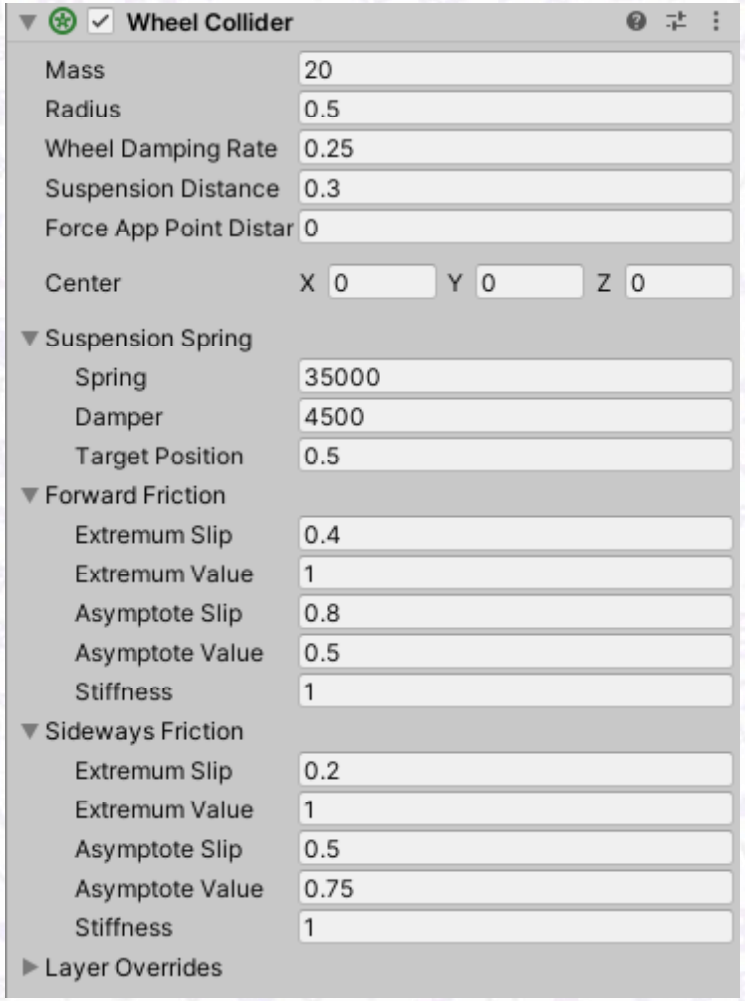
Wheel Collider

Komponent opisuje bryłę brzegową koła pojazdu.

(Samouczek: https://docs.unity3d.com/Manual/WheelColliderTutorial.html)

Uwzględnia:

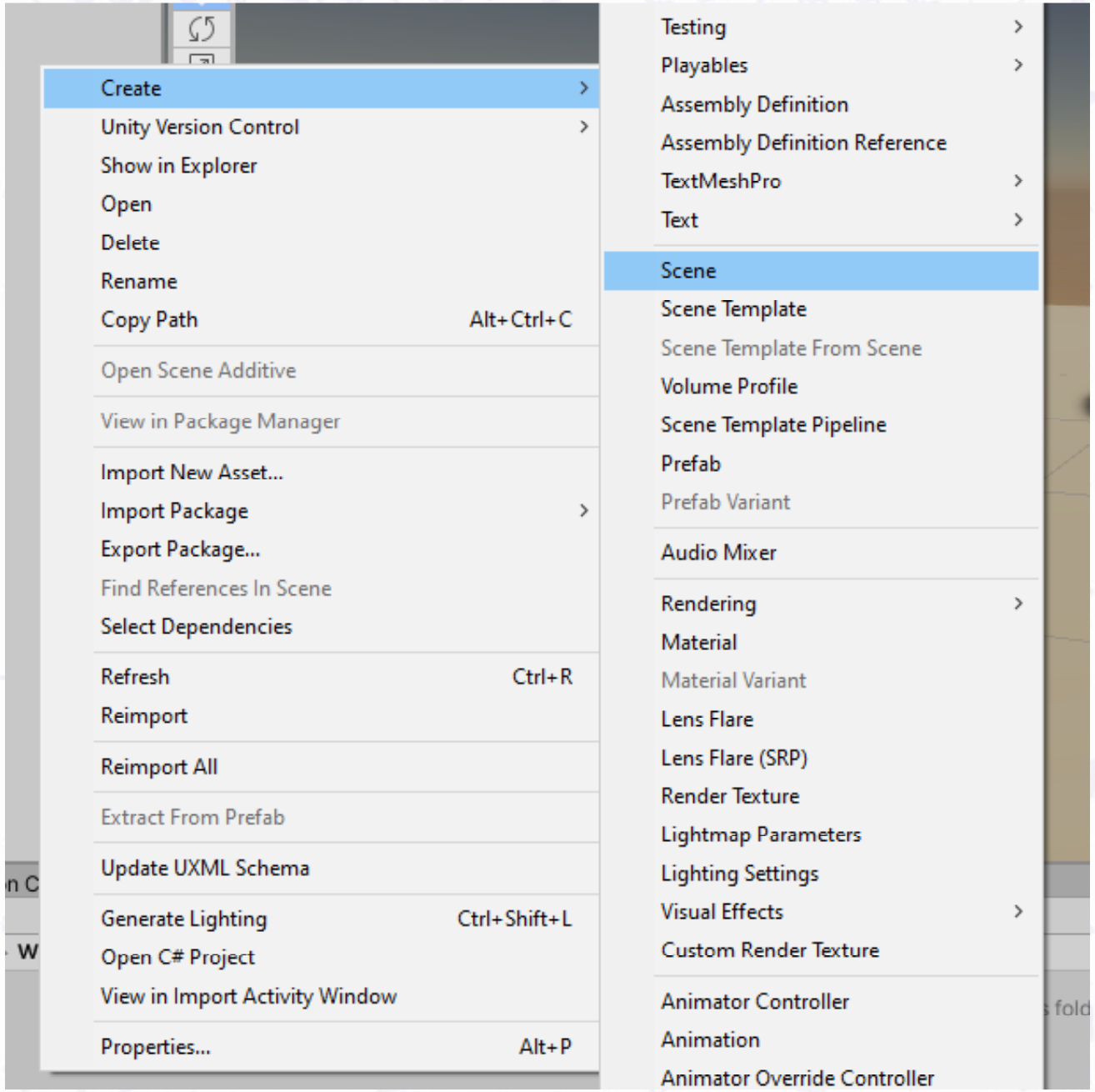
- Zderzenia z podłożem
- Fizykę obrotu koła
- Tarcie o podłoże
- Bezwładność
- Tarcie pochodzące z układu napędowego
- Sprężystość



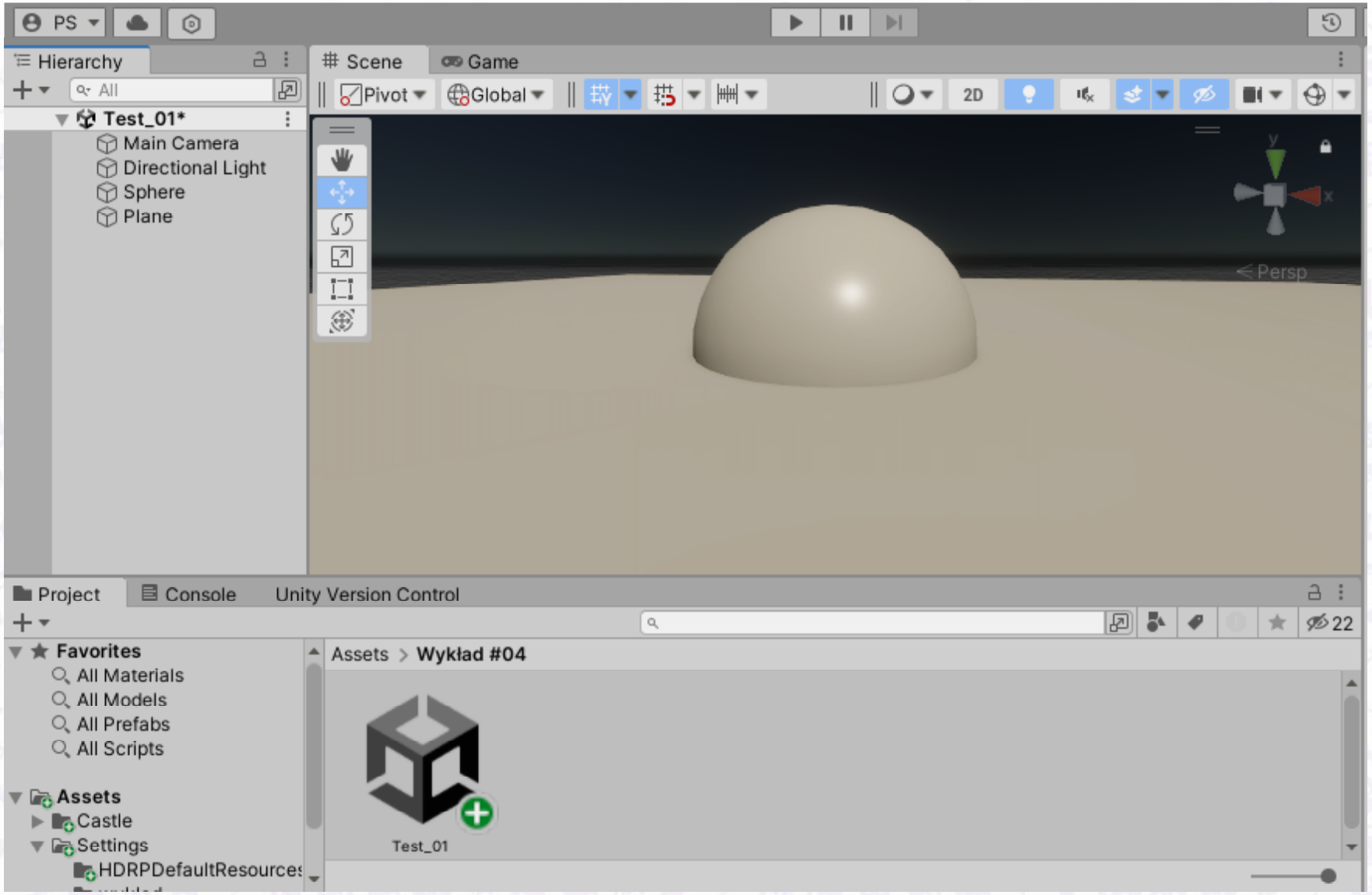
ZAAWANSOWANE TRÓJWYMIAROWE BRYŁY BRZEGOWE

Komponent	Kształt	Dodatkowe informacje
MeshCollider	Uproszczonej siatki wielokątów	Wykrywanie kolizji działa tylko z siatkami złożonymi z mniej niż 256 wielokątów wypukłych
TerrainCollider	Wynikający z mapy wysokości komponentu Terrain	Może również generować bryły brzegowe dla drzew znajdujących się na terenie
WheelCollider	Pojedynczego koła pojazdu	Dodatkowo symuluje większość fizyki związanej z działaniem koła pojazdu

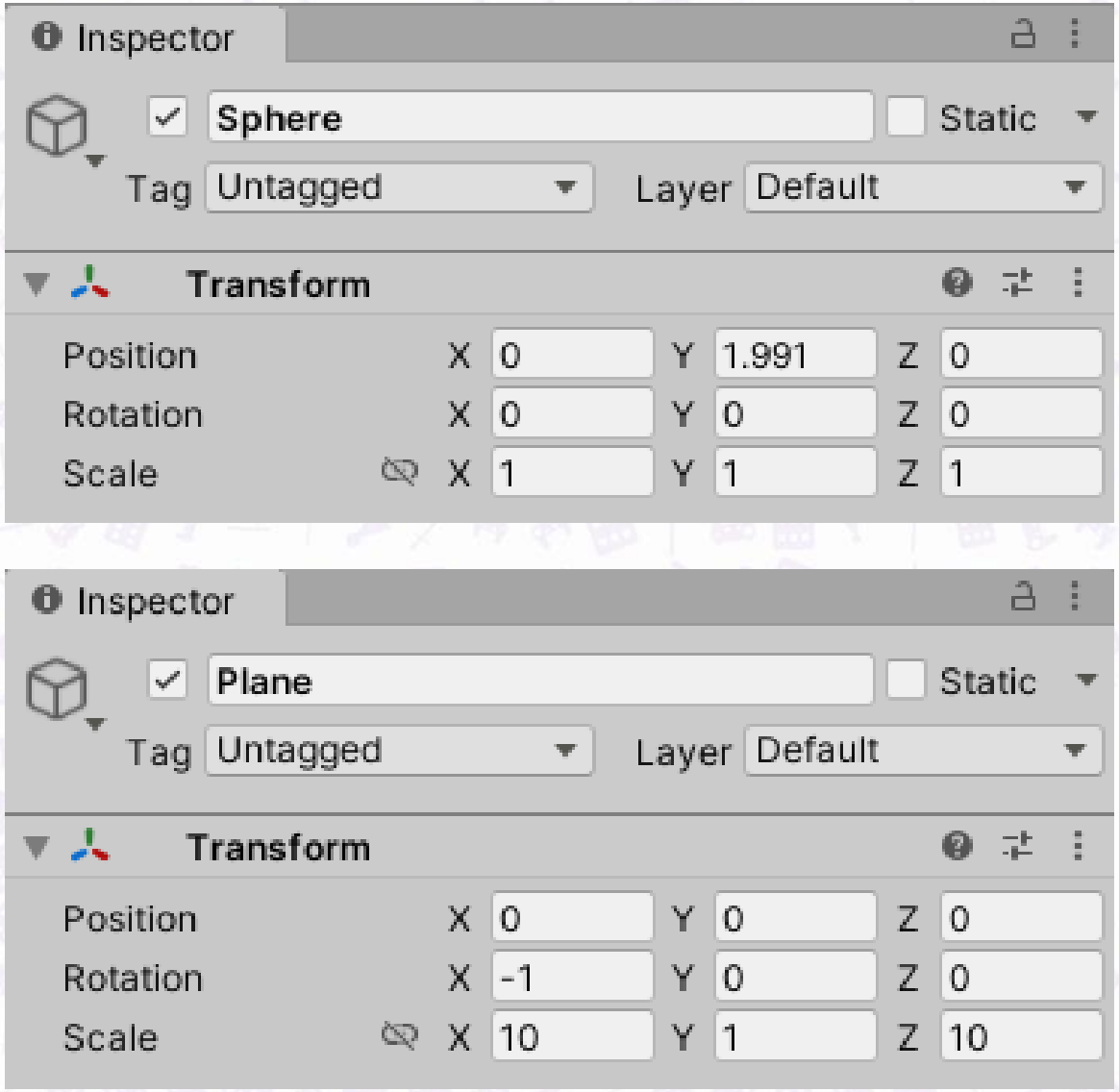
SCENA TESTOWA: TWORZENIE SCENY



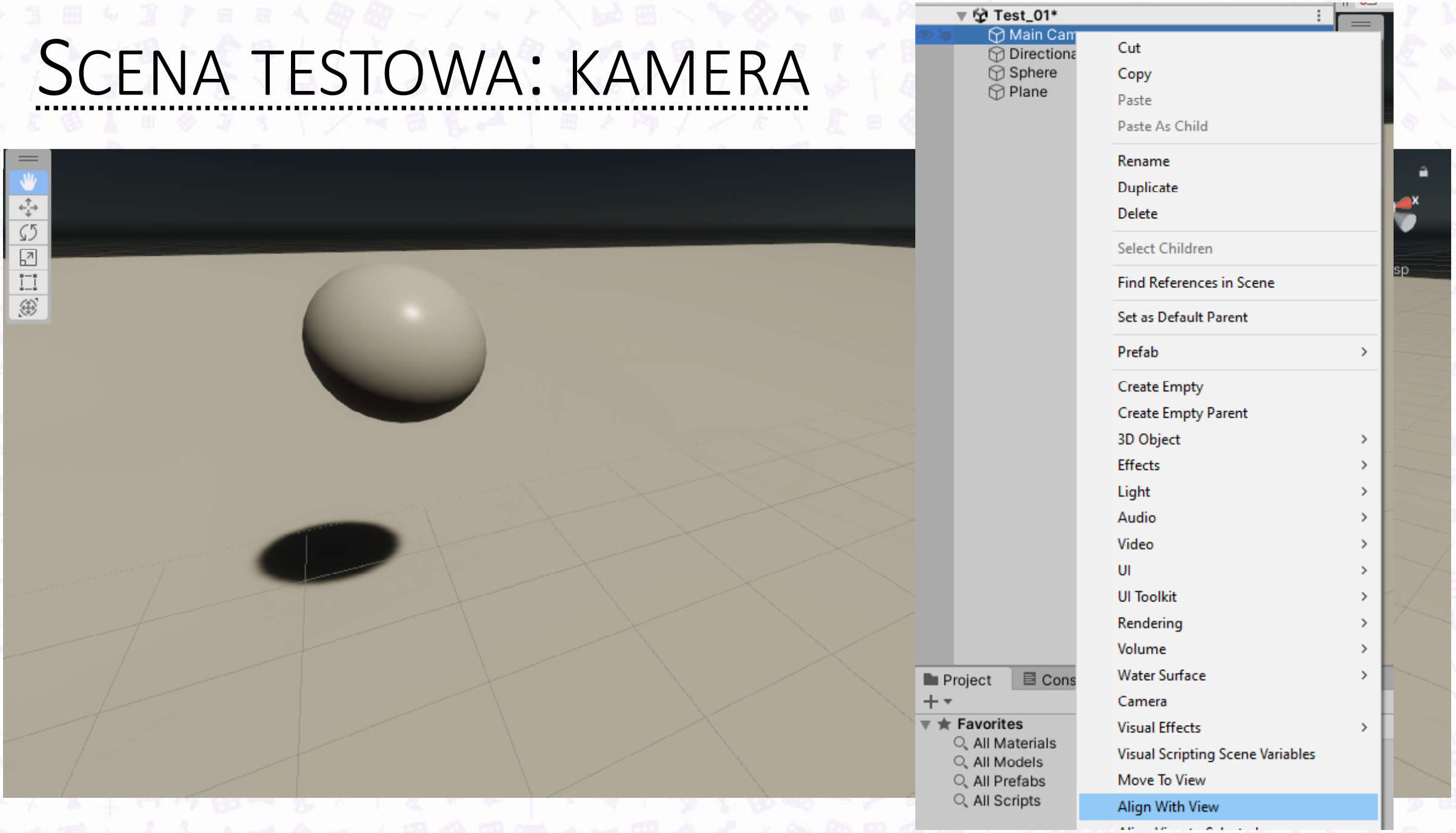
SCENA TESTOWA: DODAWANIE BYTÓW



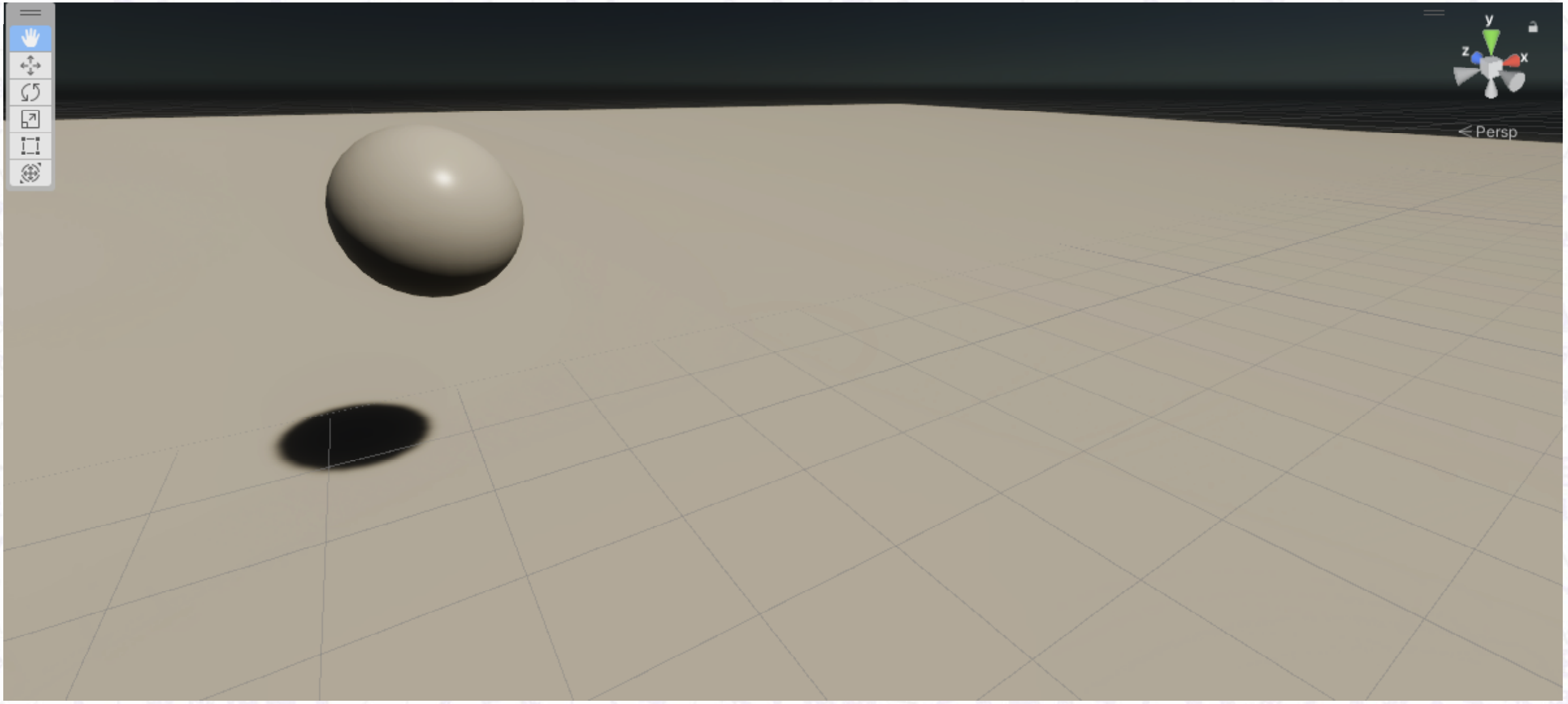
SCENA TESTOWA: ZMIANA PARAMETRÓW



SCENA TESTOWA: KAMERA



Test



RIGIDBODY

Komponent Rigidbody pozwala oznaczyć dany byt jako ciało sztywne.

Masa

Opór aerodynamiczny

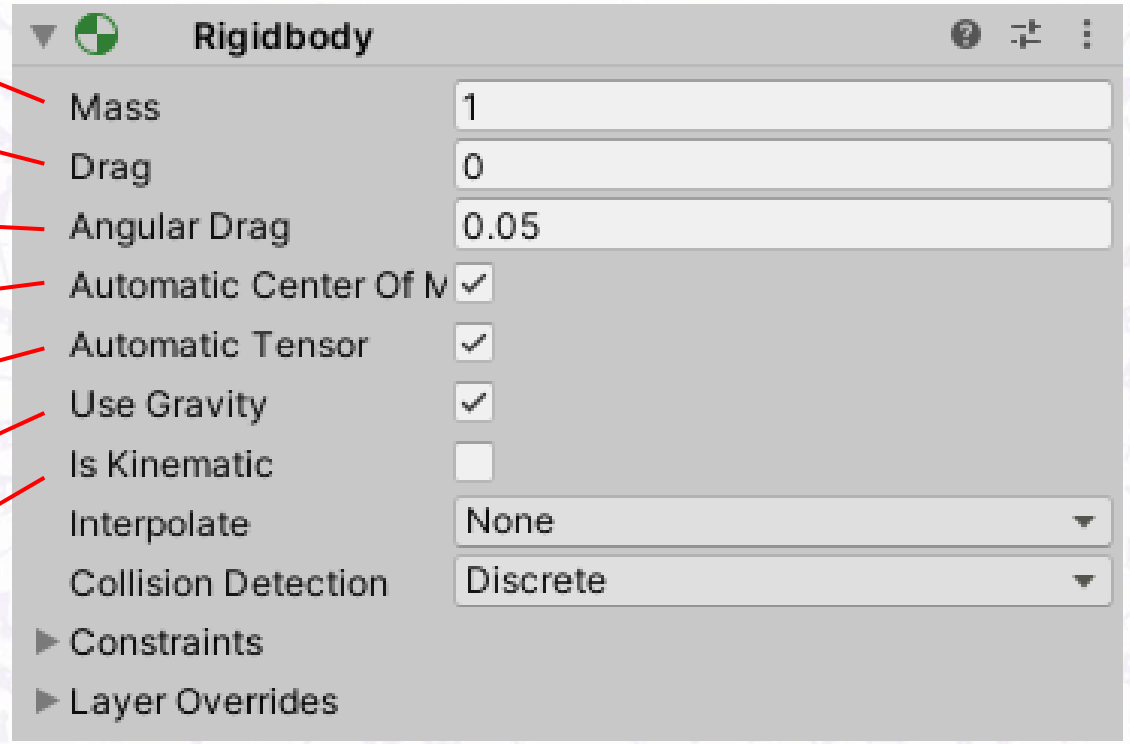
Opór kątowy

Automatyczne obliczanie centrum masy

Automatyczne obliczanie momentu bezwładności

Czy ciało reaguje na grawitację

Czy ignorować system fizyki



RIGIDBODY2D

Komponent Rigidbody pozwala oznaczyć dany byt jako ciało sztywne.

Czy ignorować system fizyki

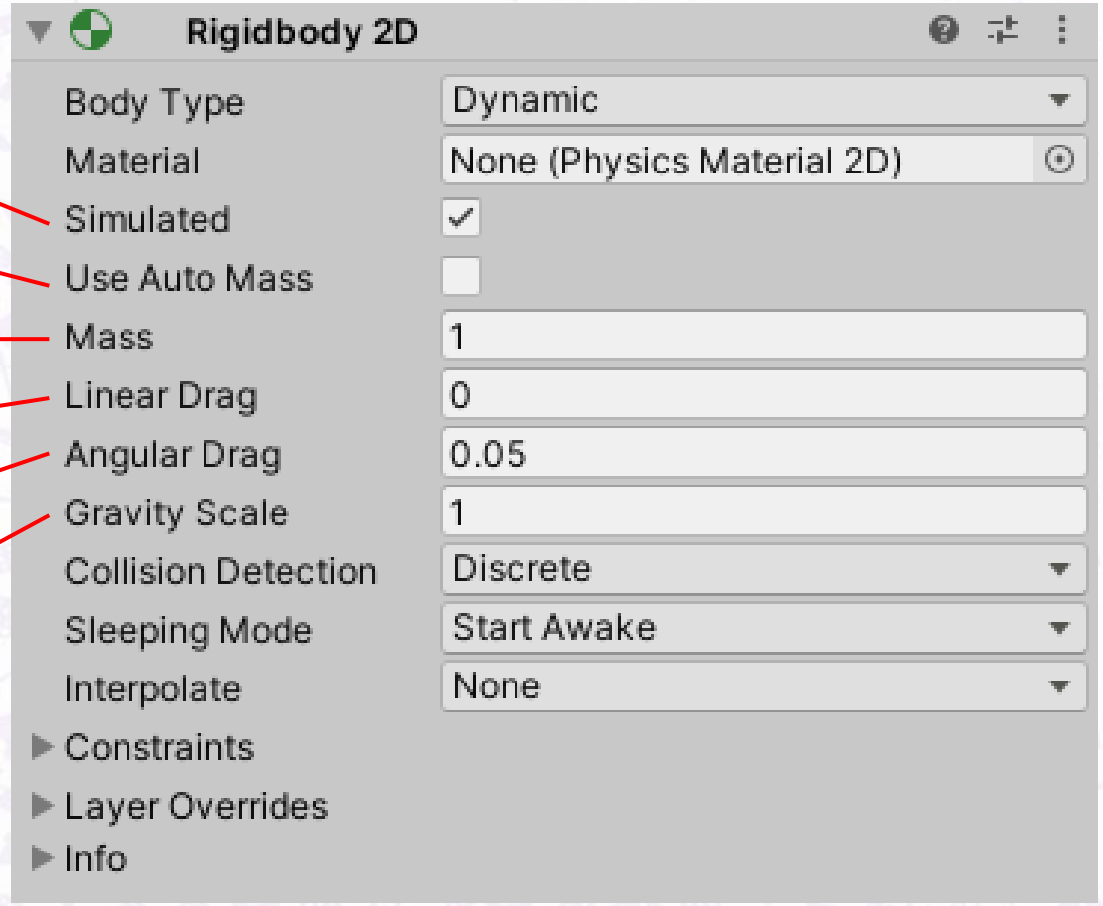
Automatyczne obliczanie centrum masy

Masa

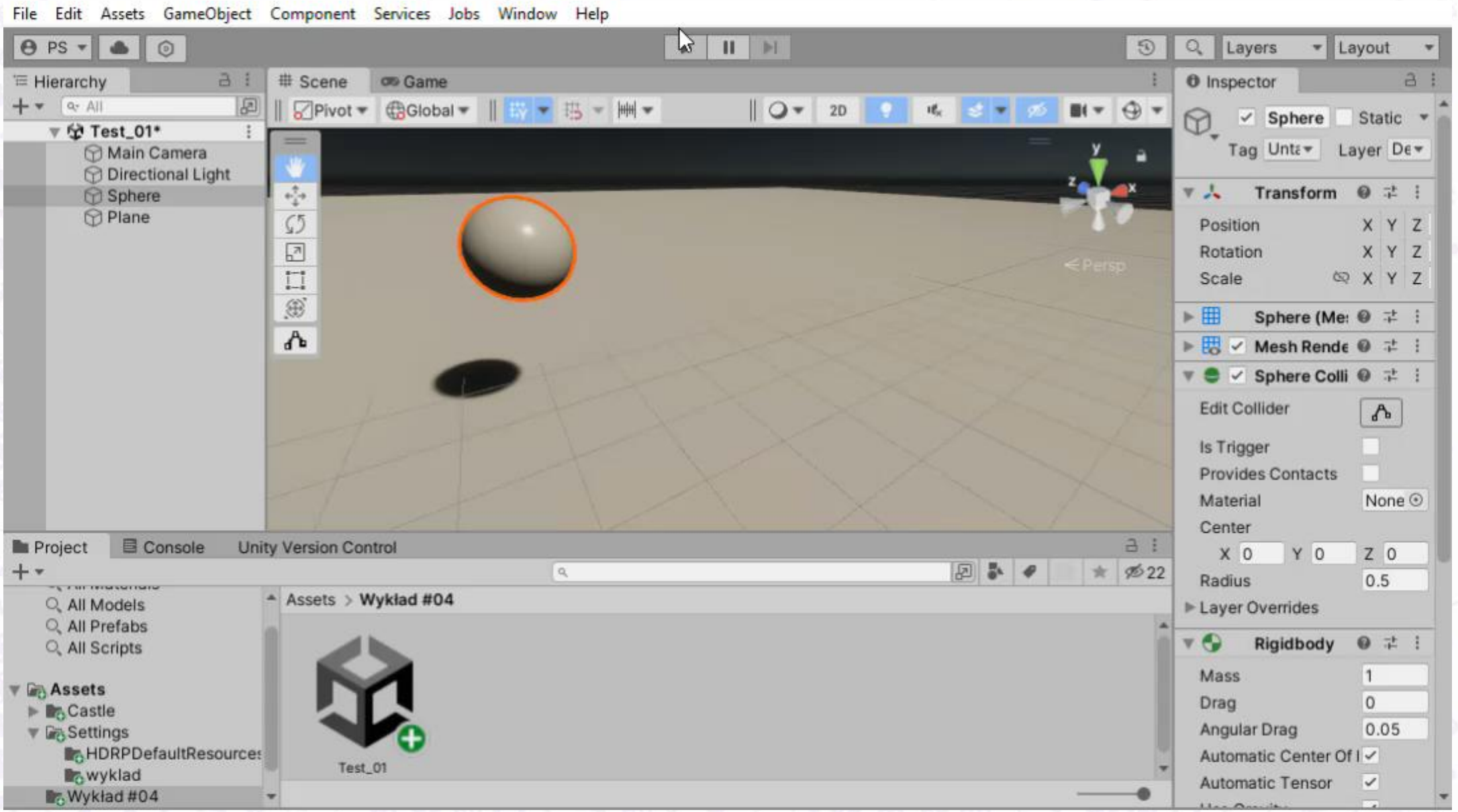
Opór aerodynamiczny

Opór kątowy

Mnożnik grawitacji



TEST - POPRAWIONY



Physic Material

Zasób określający właściwości materiału, jakim pokryty jest dany byt.

Create → 2D → Physic Material 2D

Create → Physic Material

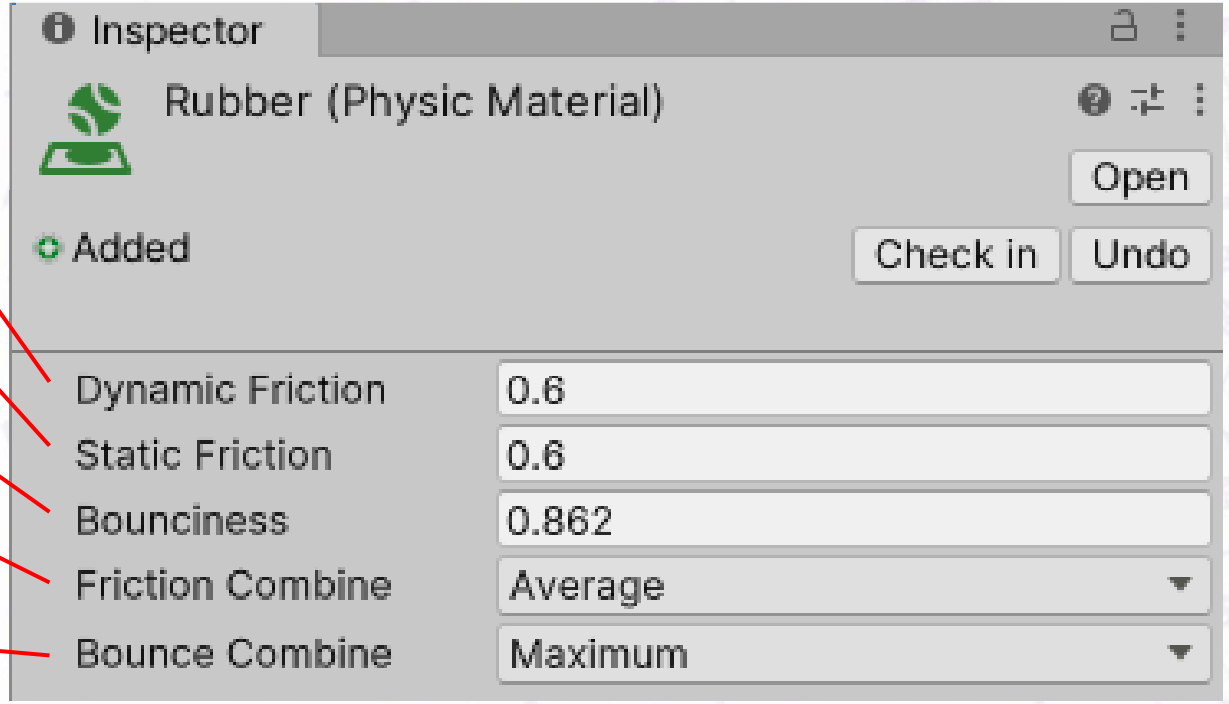
Współczynnik tarcia dynamicznego

Współczynnik tarcia statycznego

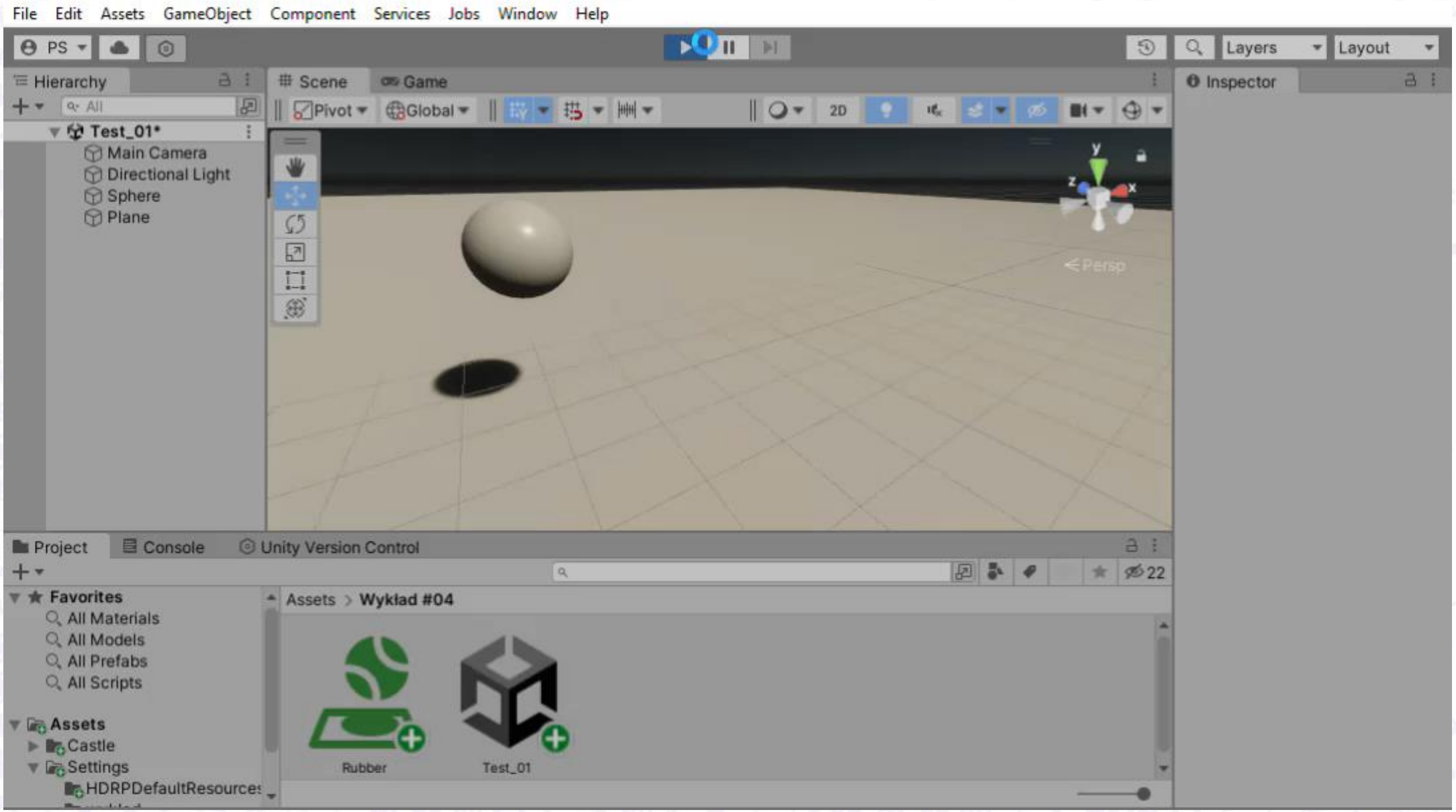
„Zdolność do odbijania się”

Funkcja obliczania tarcia

Funkcja obliczania odbijania



TEST - ODBIJANIE



DOKUMENTACJA/PISANIE SKRYPTÓW

Unity Documentation | Manual | Scripting API | Search scripting...

Version: 2022.3 | C#

- ResourceRequest
- Resources
- ResourcesAPI
- Rigidbody
- **Rigidbody2D**
- RuntimeAnimatorController
- ScalableBufferManager
- Screen
- ScreenCapture
- ScriptableObject
- SecondarySpriteTexture
- Security
- Shader
- ShaderVariantCollection
- SkeletonBone
- SkinnedMeshRenderer
- Skybox
- SleepTimeout
- SliderJoint2D
- Snapping
- Social
- SoftJointLimit
- SoftJointLimitSpring
- SortingLayer
- SparseTexture

Rigidbody2D

class in UnityEngine / Inherits from:Component/
Implemented in:UnityEngine.Physics2DModule

Leave feedback

SWITCH TO MANUAL

Description

Rigidbody physics component for 2D sprites.

The Rigidbody2D class essentially provides the same functionality in 2D that the Rigidbody class provides in 3D. Adding a Rigidbody2D component to a sprite puts it under the control of the physics engine. By itself, this means that the sprite will be affected by gravity and can be controlled from scripts using forces. By adding the appropriate collider component, the sprite will also respond to collisions with other sprites. This behaviour comes entirely from Unity's physics system; very little code is required to get impressive and authentic physical behaviour and allows for "emergent" gameplay that was not explicitly coded into the game.

Additional resources: Rigidbody class, SpriteRenderer class, Collider2D class, Joint2D class.

Properties

angularDrag	Coefficient of angular drag.
angularVelocity	Angular velocity in degrees per second.
attachedColliderCount	Returns the number of Collider2D attached to this Rigidbody2D.
bodyType	The physical behaviour type of the Rigidbody2D.
centerOfMass	The center of mass of the rigidBody in local space.

PROBLEMY PRAKTYCZNE

Problem 1: wykrywanie kolizji z poziomu skryptów

Funkcje klasy Collider

Funkcja	Moment wywołania
OnCollisionEnter	Jednorazowo, w klatce, w której dana bryła brzegowa wejdzie w kolizję z inną bryłą brzegową lub ciałem sztywnym.
OnCollisionStay	**Co klatkę**, dopóki dana bryła brzegowa koliduje z inną bryłą brzegową lub ciałem sztywnym.
OnCollisionExit	Jednorazowo, w klatce, w której dana bryła brzegowa zakończy kolidować z inną bryłą brzegową lub ciałem sztywnym.
OnTriggerEnter	Jednorazowo, w klatce, w której inna bryła brzegowa wejdzie w kolizję z wyzwalaczem opisanym przez tę bryłę brzegową.
OnTriggerStay	**Prawie w każdej klatce**, dopóki inna bryła brzegowa koliduje z wyzwalaczem opisanym przez tę bryłę brzegową.
OnTriggerExit	Jednorazowo, w klatce, w której inna bryła brzegowa zakończy stykać się z wyzwalaczem opisanym przez tę bryłę brzegową.

Wykrywanie kolizji: skrypt

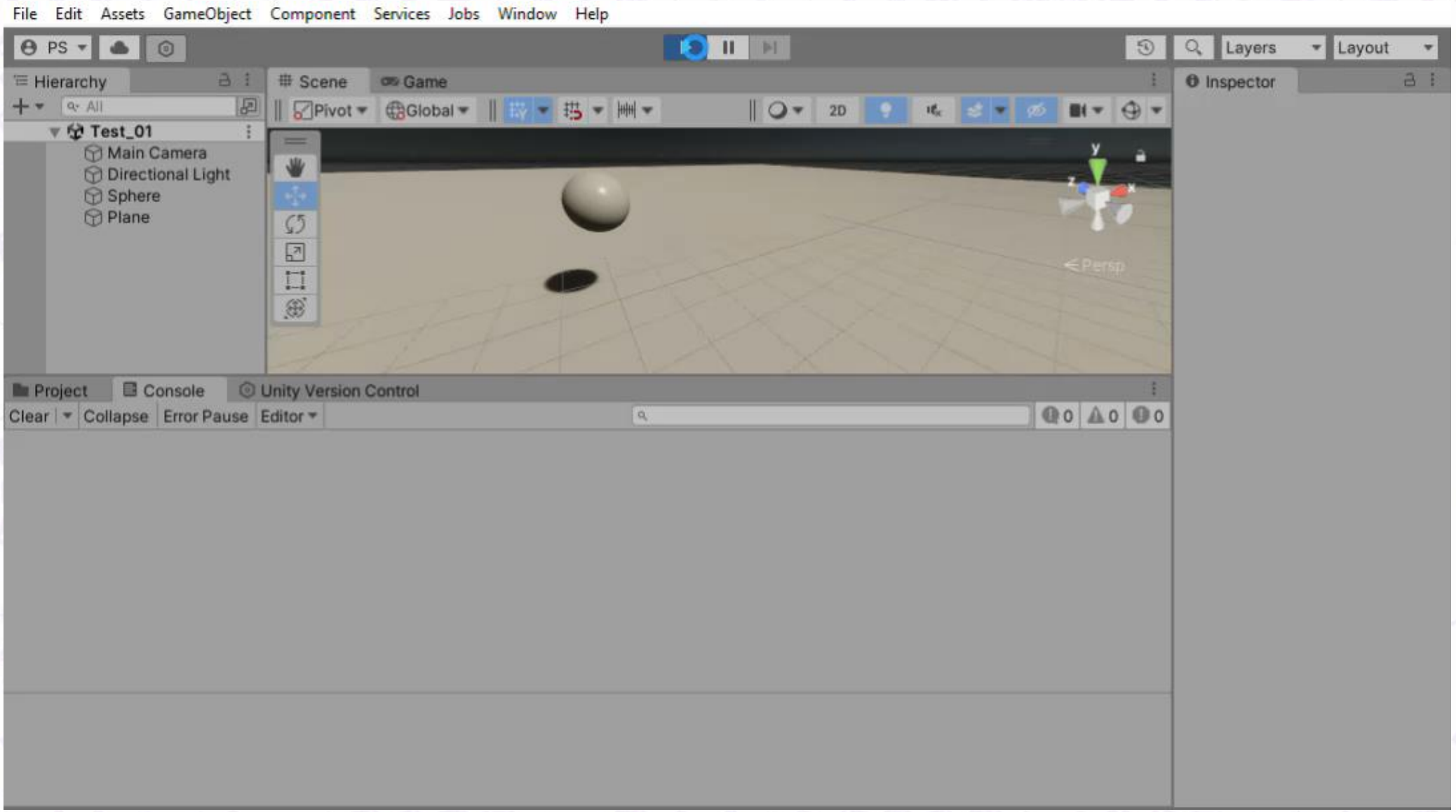
```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;

4. public class CollisionDetection : MonoBehaviour
5. {
6.     private void OnCollisionEnter(Collision collision)
7.     {
8.         Debug.Log("Kontakt!");
9.     }

10.    private void OnCollisionStay(Collision collision)
11.    {
12.        Debug.Log("Utrzymuje kontakt.");
13.    }

14.    private void OnCollisionExit(Collision collision)
15.    {
16.        Debug.Log("Odbite.");
17.    }
18.}
```

WYKRYWANIE KOLIZJI



PROBLEMY PRAKTYCZNE

Problem 2: odczytanie kontekstu kolizji

ROZPOZNAWANIE KONTEKSTU: SKRYPT

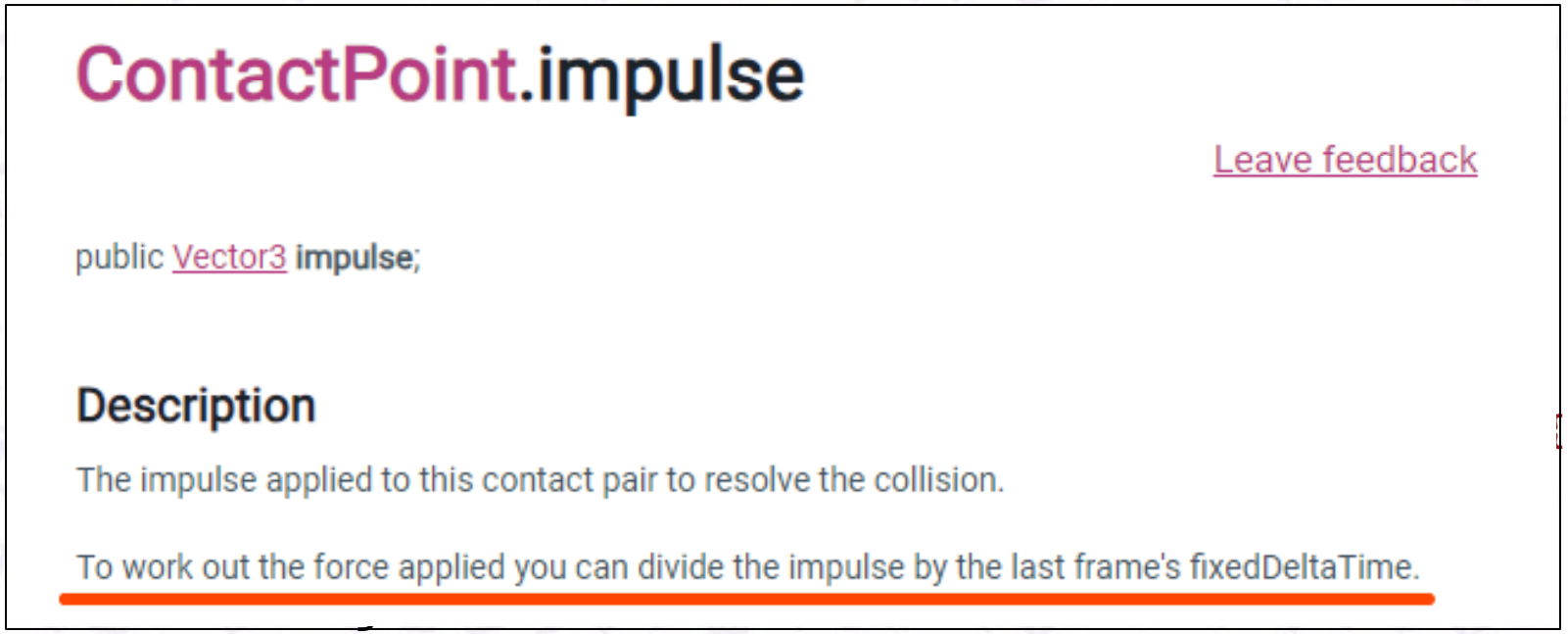
```
1.  private void OnCollisionEnter(Collision collision)
2.      {
3.          GameObject bytKolidujacy = collision.gameObject;
4.          string nazwaObiektu = bytKolidujacy.name;
5.          ContactPoint miejsceZderzenia = collision.GetContact(0);
6.          float energia = miejsceZderzenia.impulse.magnitude;
7.          Debug.Log(string.Format("Kontakt z {0} w punkcie {1} z siłą {2}!",
8.              nazwaObiektu,
9.              miejsceZderzenia.point,
10.             energia)
11.             );
12.     }
```

Kontakt z Plane w punkcie (0.00, -0.06, -2.87) z siłą 4,656351!

PROBLEMY PRAKTYCZNE

Problem 3: uwzględnienie upływu czasu

ROZPOZNAWANIE KONTEKSTU: SKRYPT



ContactPoint.impulse

Leave feedback

public Vector3 impulse;

Description

The impulse applied to this contact pair to resolve the collision.

To work out the force applied you can divide the impulse by the last frame's fixedDeltaTime.

```
1.
2.
3.
4.
5.
6.
7.                                                    2}!",
8.
9.
10.
11.             );
12.     }
```

Co to za wartość?

Kontakt z Plane w punkcie (0.00, -0.06, -2.87) z siłą 4,656351!

TIME

Klasa pozwalająca uzyskać wszelkie informacje związane z upływem czasu

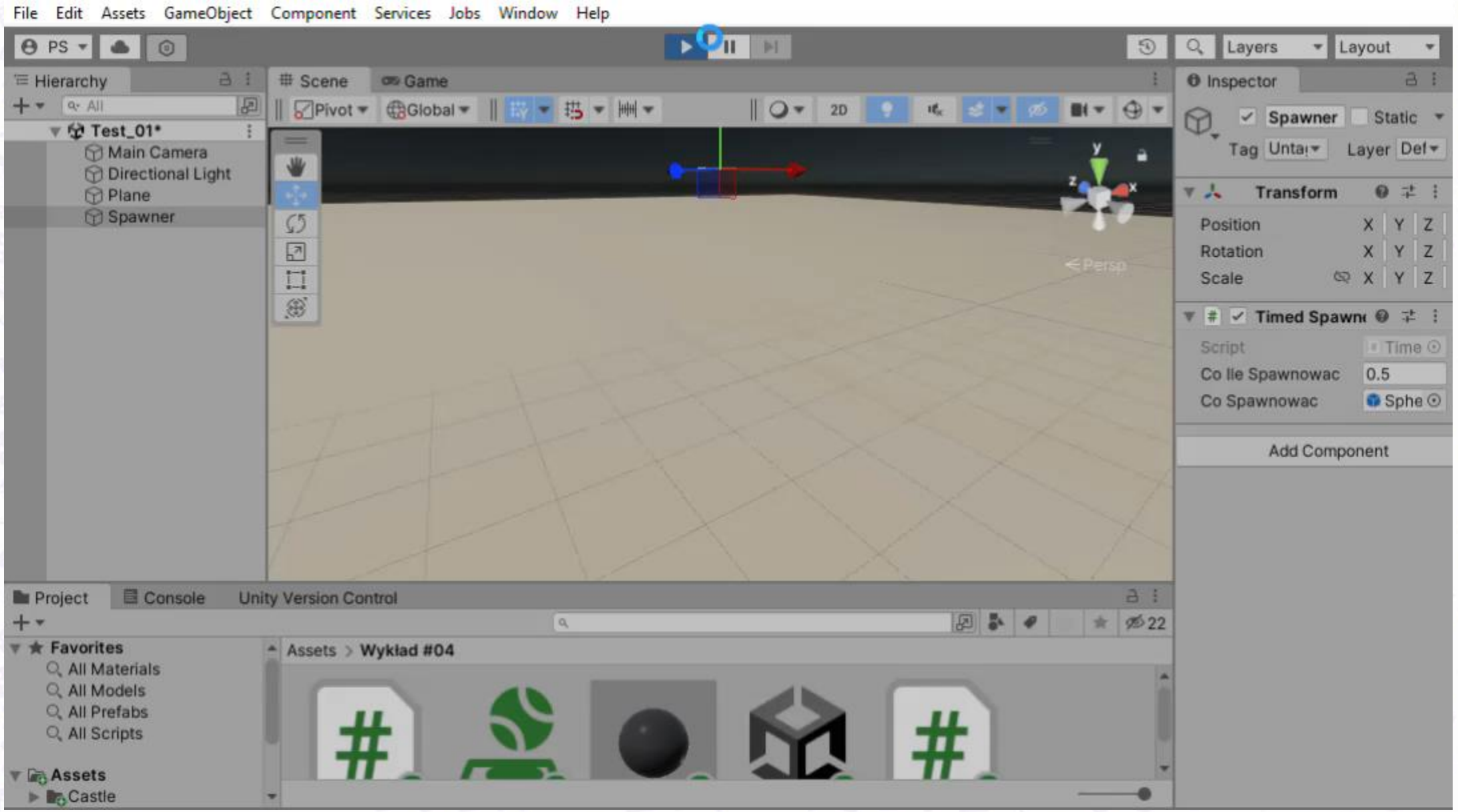
https://docs.unity3d.com/ScriptReference/Time.html

Pole	Przechowywana wartość
time	Czas jaki upłynął od uruchomienia aplikacji, aż do początku aktualnej klatki animacji (Update), podany w sekundach.
fixedTime	Czas jaki upłynął od uruchomienia aplikacji, aż do początku aktualnej klatki obliczeń fizycznych (FixedUpdate), podany w sekundach.
deltaTime	Czas jaki upłynął od początku poprzedniej klatki animacji, do początku aktualnej klatki animacji, podany w sekundach
fixedDeltaTime	Czas jaki upłynął od początku poprzedniej klatki obliczeń fizycznych, do początku aktualnej klatki obliczeń fizycznych, podany w sekundach (**STAŁY!**)

Upływ czasu: skrypt

```
1. public class TimedSpawner : MonoBehaviour
2. {
3.     public float coIleSpawnowac = 0.5f;
4.     public GameObject coSpawnowac = null;
5.
6.     private float ileUplynelo = 0f;
7.
8.     // Update is called once per frame
9.     void Update()
10.    {
11.        ileUplynelo += Time.deltaTime;
12.        if(ileUplynelo >= coIleSpawnowac)
13.        {
14.            ileUplynelo = 0f;
15.            Instantiate(coSpawnowac, transform.position, transform.rotation);
16.        }
17.    }
18.}
```

OKRESOWE TWORZENIE PREFABRYKATÓW



Dziękuję za uwagę

Proszę o pytania