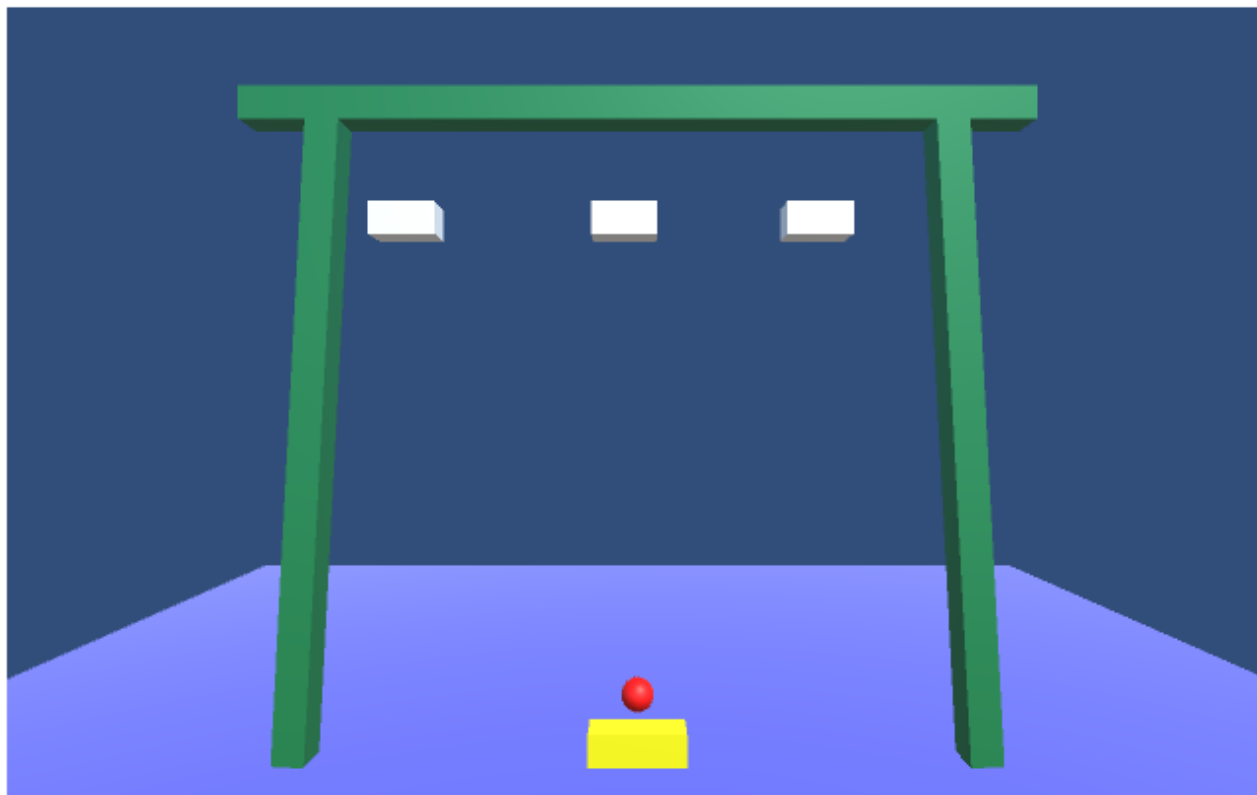


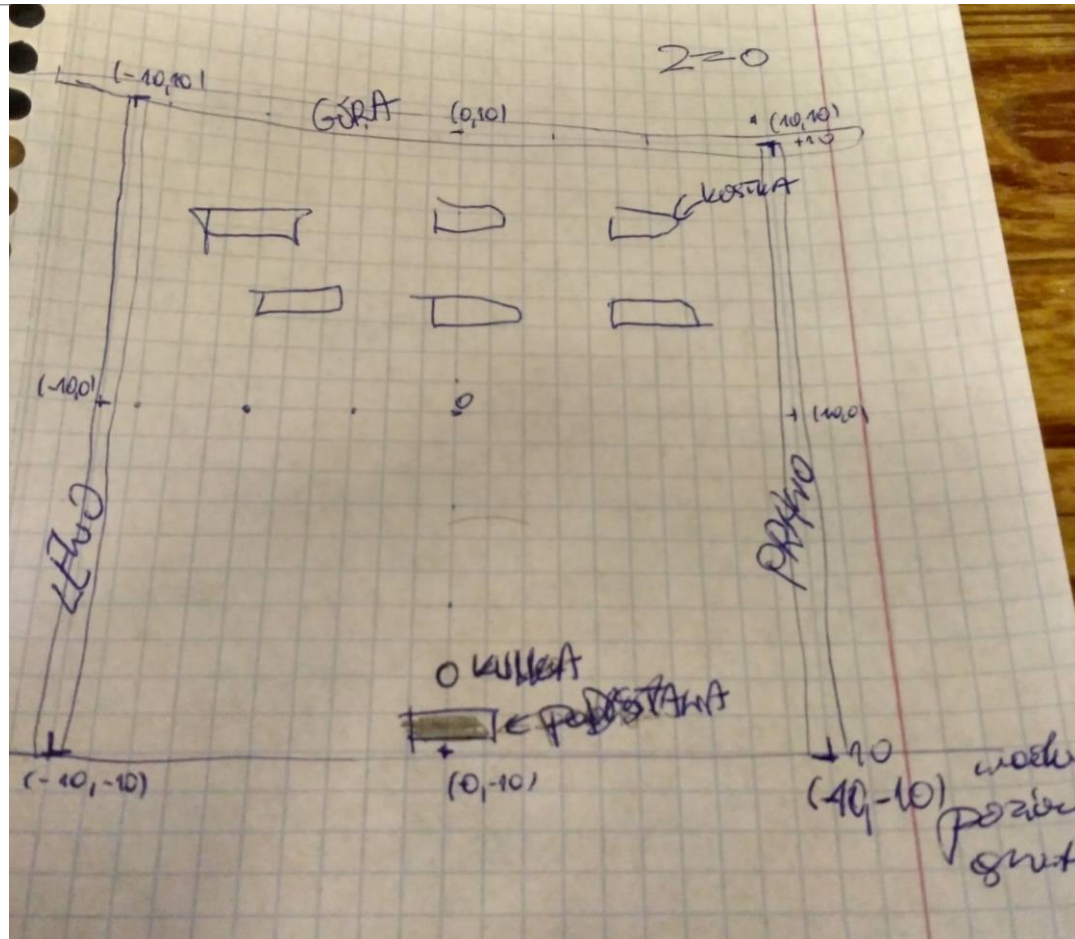
Gra Breakout

- nauka skryptów i kolizji

Co chcemy zrobić?



Potrzebny plan!



Startowe obiekty gry:

Podstawa

Lewo, Prawo, Góra – trzy ściany

Woda – poziom gruntu

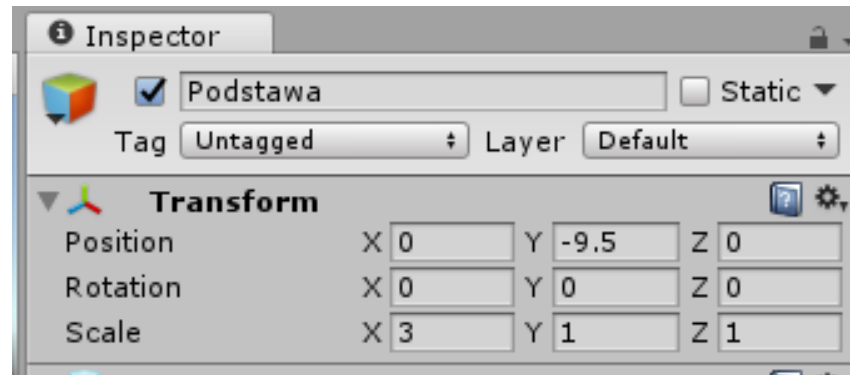
Kostka1, Kostka2, Kostka3, – elementy do zbijania

Kulka – nasz „gracz”

Ważne założenie: mimo, że tworzymy grę 3D zakładamy, że wszystkie obiekty mają współrzędną z równo zero (w pierwszej pozycji od góry).

Podstawa

Typ: Cube



Ściany, typ: Cube

Lewo

Transform						
Position	X	-10	Y	0	Z	0
Rotation	X	0	Y	0	Z	-3
Scale	X	1	Y	20	Z	1

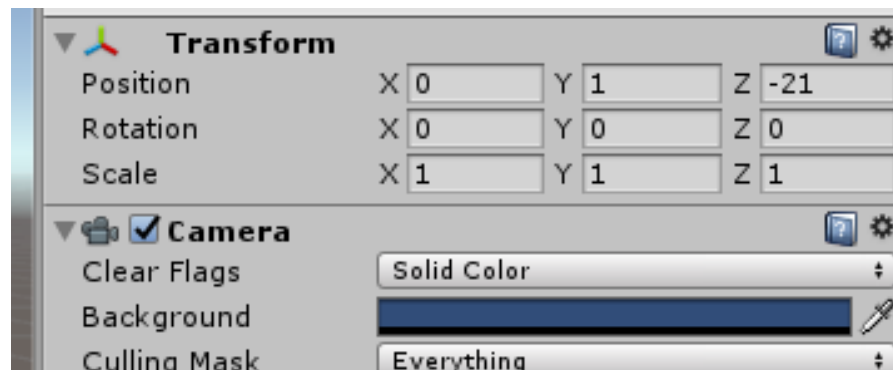
Prawo

Transform						
Position	X	10	Y	0	Z	0
Rotation	X	0	Y	0	Z	3
Scale	X	1	Y	20	Z	1

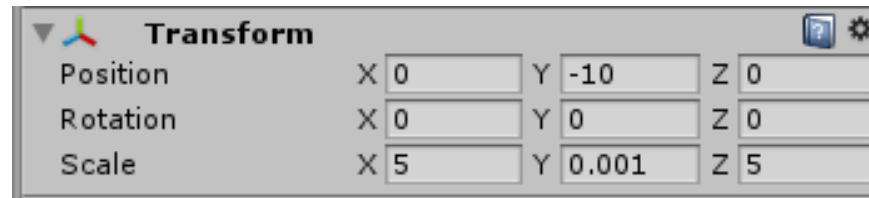
Góra

Transform						
Position	X	0	Y	10	Z	0
Rotation	X	0	Y	0	Z	90
Scale	X	1	Y	24	Z	1

Ustawienia kamery



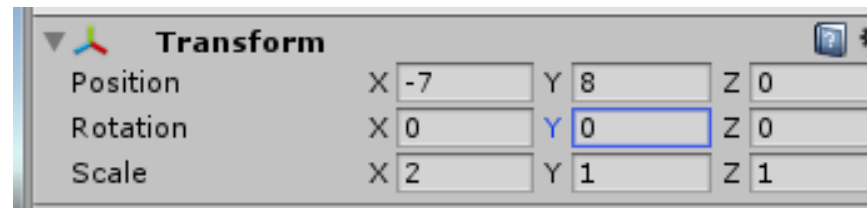
Woda, typ Plane



A screenshot of a software interface panel titled "Transform". The panel has a dropdown arrow and a small 3D coordinate icon on the left, and a help icon and a settings gear icon on the right. It contains three rows of controls: "Position", "Rotation", and "Scale". Each row has three input fields labeled X, Y, and Z. The values are: Position (X: 0, Y: -10, Z: 0), Rotation (X: 0, Y: 0, Z: 0), and Scale (X: 5, Y: 0.001, Z: 5).

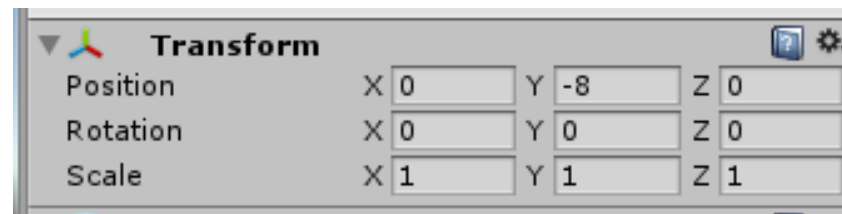
	X	Y	Z
Position	0	-10	0
Rotation	0	0	0
Scale	5	0.001	5

Kostka1, typ Cube



I kilka podobnie....

Kulka, typ: Sphere



A screenshot of a software interface's 'Transform' panel. The panel has a title bar with a dropdown arrow, a small 3D coordinate icon, the word 'Transform', a help icon (question mark in a blue box), and a settings icon (gear). Below the title bar, there are three rows of controls: 'Position', 'Rotation', and 'Scale'. Each row has three input fields labeled 'X', 'Y', and 'Z'. The 'Position' row has values 0, -8, and 0. The 'Rotation' row has values 0, 0, and 0. The 'Scale' row has values 1, 1, and 1.

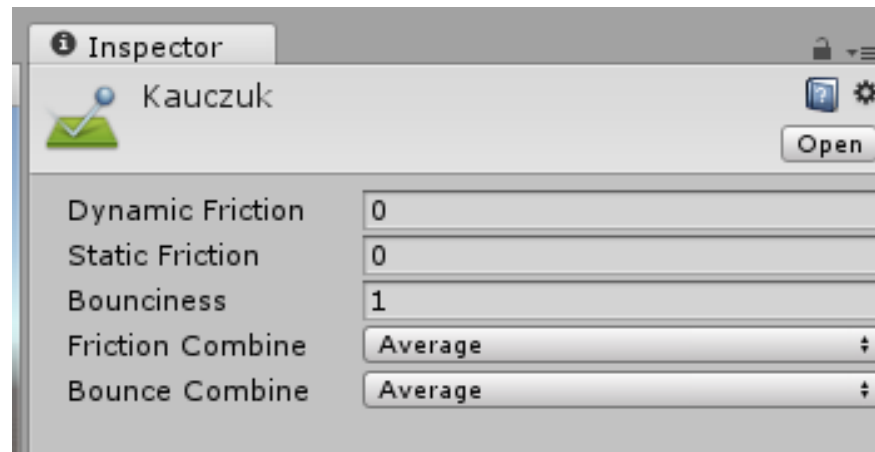
	X	Y	Z
Position	0	-8	0
Rotation	0	0	0
Scale	1	1	1

Kolorujemy!

Tworzymy kilka materiałów i przypisujemy obiektom kolory.

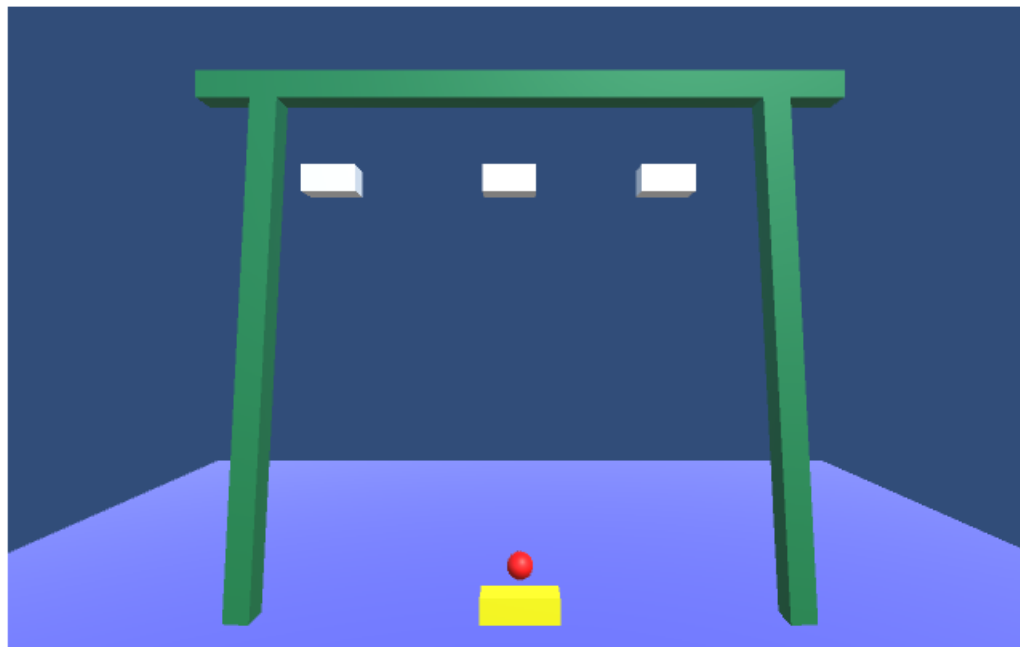
Dodajemy fizyczny materiał!

Tworzymy fizyczny materiał Kauczuk tak, aby w całości przekazywał energię kinetyczną.



Plan na skrypty

Co będzie się poruszać?



Skrypt - Podstawa

- Dodajmy do podstawy komponent Rigidbody, wyłączmy grawitację i włączmy opcję „Is Kinematic” (dobra praktyka, a nie konieczność)

`Vector3(x,y,z)` – trójwymiarowy wektor o współrzędnych (x,y,z).

`transform.position.x` – pobranie pozycji x z zakładki Transform aktualnego obiektu

`Input.GetAxis("Horizontal")` – „liczba ruchów” w kierunku poziomym

`Mathf.Clamp(a,b,c)` = a jeśli $b \leq a \leq c$

= b jeśli $a < b$

= c jeśli $a > c$

Skrypt - Kulka

Dodajemy Rigidbody – bez grawitacji i „Is Kinematic”.

`GetComponent<Rigidbody>()` – pobranie komponentu rigidbody

`Input.GetButtonDown("Fire1")` – zwraca true jeśli ktoś nacisnął klawisz

`rb.AddForce` – dodanie „siły” do bryły sztywnej

`velocity` – ustawienie „szybkości”

`gameObject.SetActive` – ustawienie aktywności

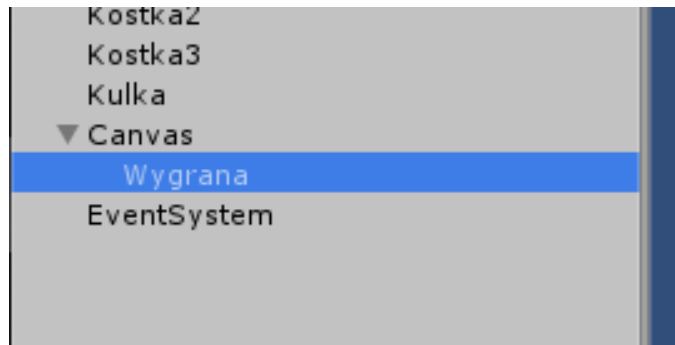
`Vector3.zero` – wektor zerowy (0,0,0)

`gameObject.CompareTag` – sprawdzenie tagu

Jak gra ma się kończyć?

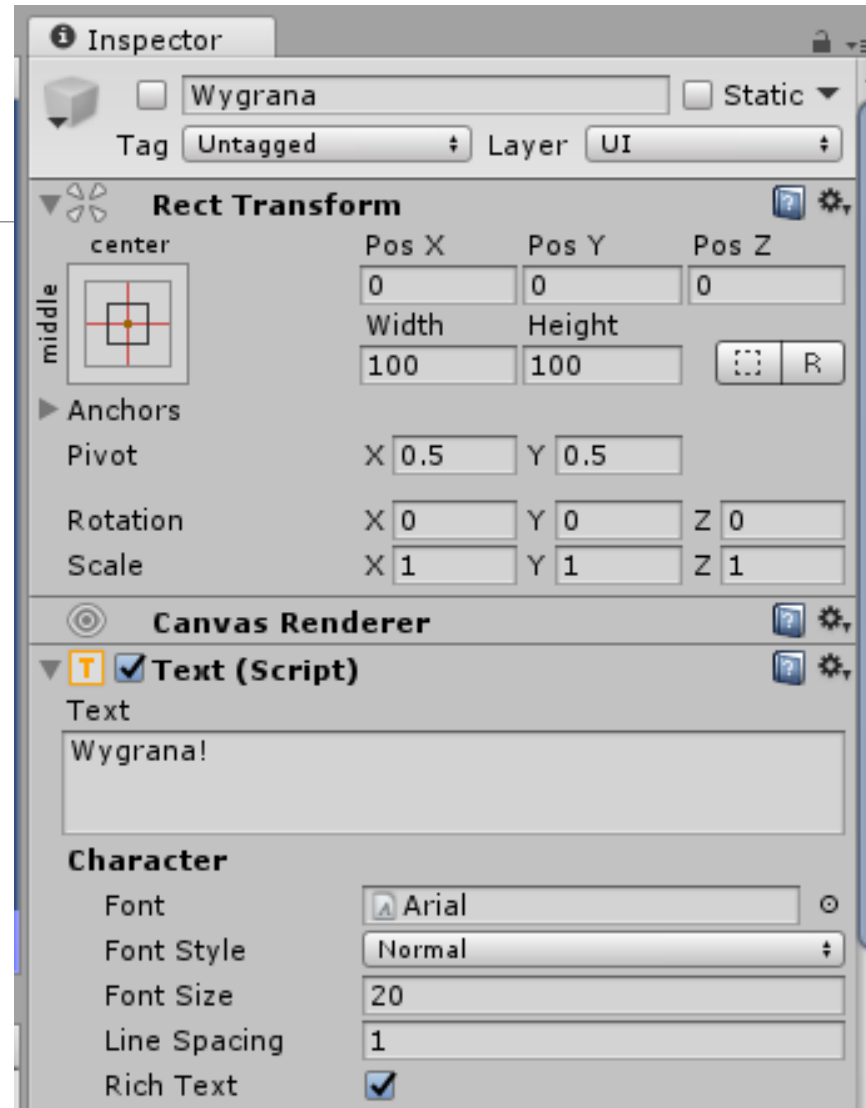
- ☐ Dodamy napis Wygrana!
- ☐ Ukryjemy kulkę.

Text - Wygrana



Ważne: trzeba dodać
bibliotekę w skrypcie

using UnityEngine.UI;

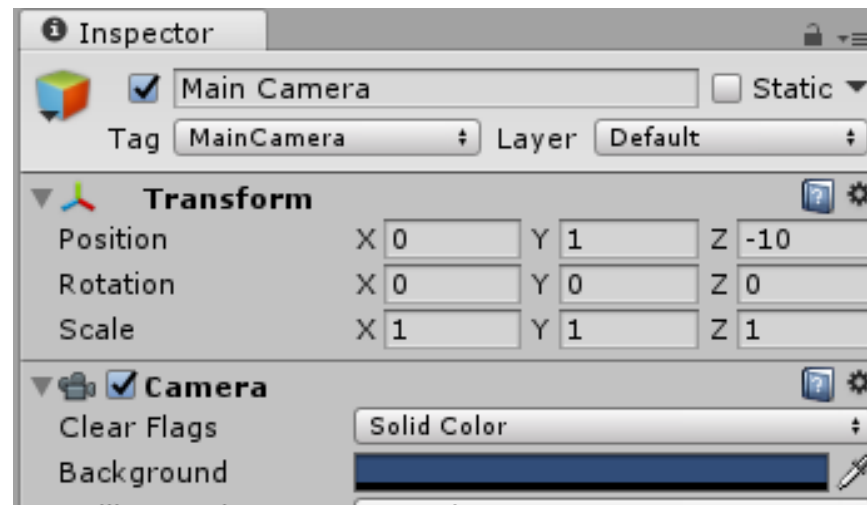


Zapisanie sceny – Level1

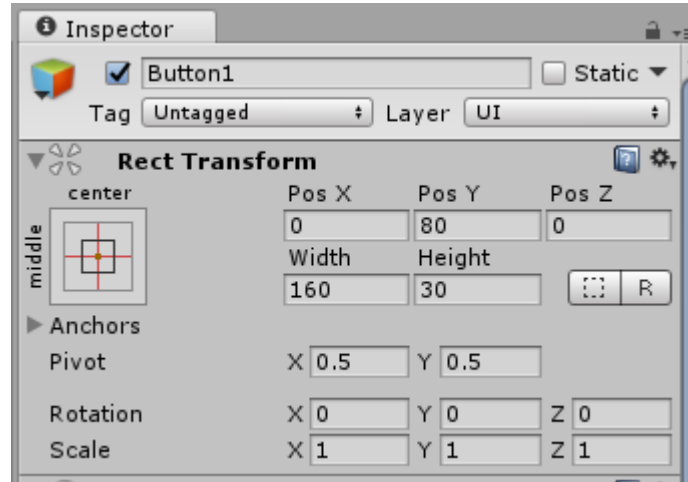
Co dalej?

Nowa scena - Menu

Ustawmy najpierw kamerę



Dodajemy kilka Buttonów



Skrypt - MenuSkrypt

Tworzy pusty obiekt MenuSkrypt i dodajemy do niego skrypt MenuSkrypt.

`btn.onClick.AddListener(WłasnaMetoda);` - dodanie metody do przycisku

`using UnityEngine.SceneManagement;`
`SceneManager.LoadScene("Scena1");` - przejście do innej sceny

`Application.Quit();` - wyjście z aplikacji