F44 RED GAME JAM W DĄBROWIE GÓRNICZEJ

ORGANIZATORZY: KOŁO NAUKOWE F44 RED W CIESZYNIE

PODSTAWY UNITY

AUTOR: DAWID RASZKA

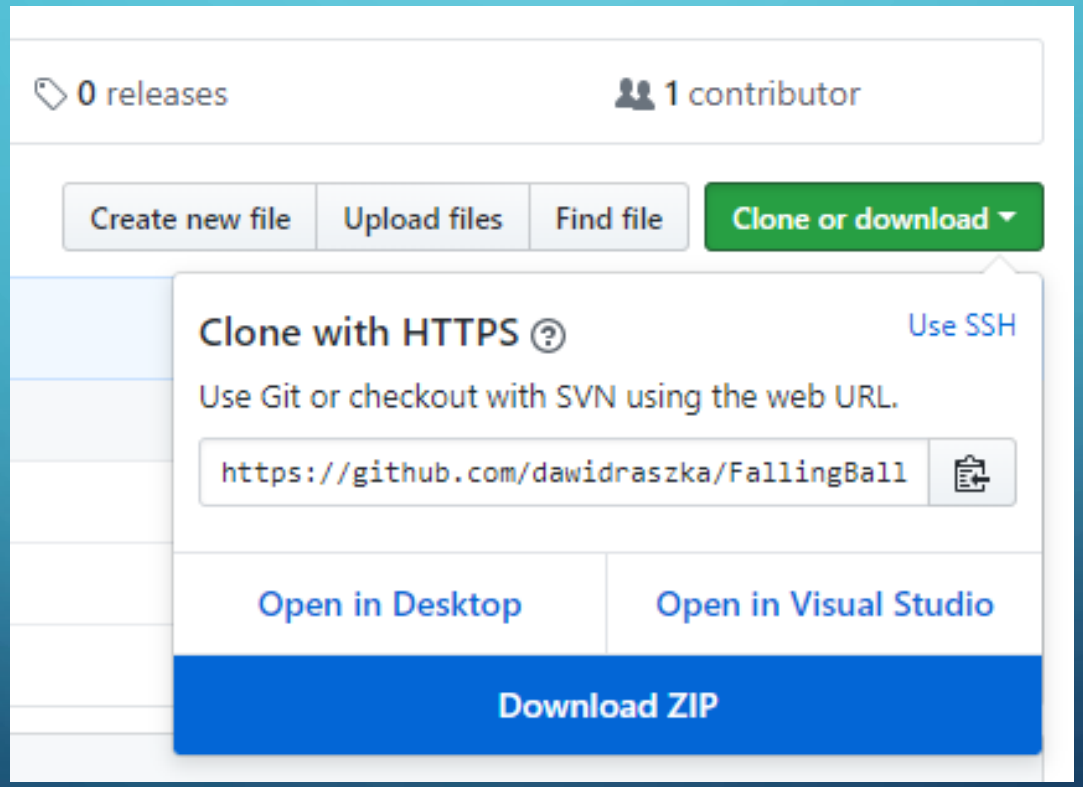
CO WŁAŚCIWIE CHCEMY ZROBIĆ, CZYLI POKAZ GOTOWEJ GRY

TEMATYKA PREZENTACJI

- Pobranie projektu
- Ustawianie sceny i poruszanie się po niej
- Kolizje oraz podstawowa fizyka
- Dodawanie skryptów, podstawowe funkcje
- Warstwy i tagi
- Przeszkody w grze, prefaby
- Poruszanie kamery
- Tworzenie obiektów z poziomu kodu
- Usuwanie obiektów z poziomu kodu
- Przesuwające się tło
- UI

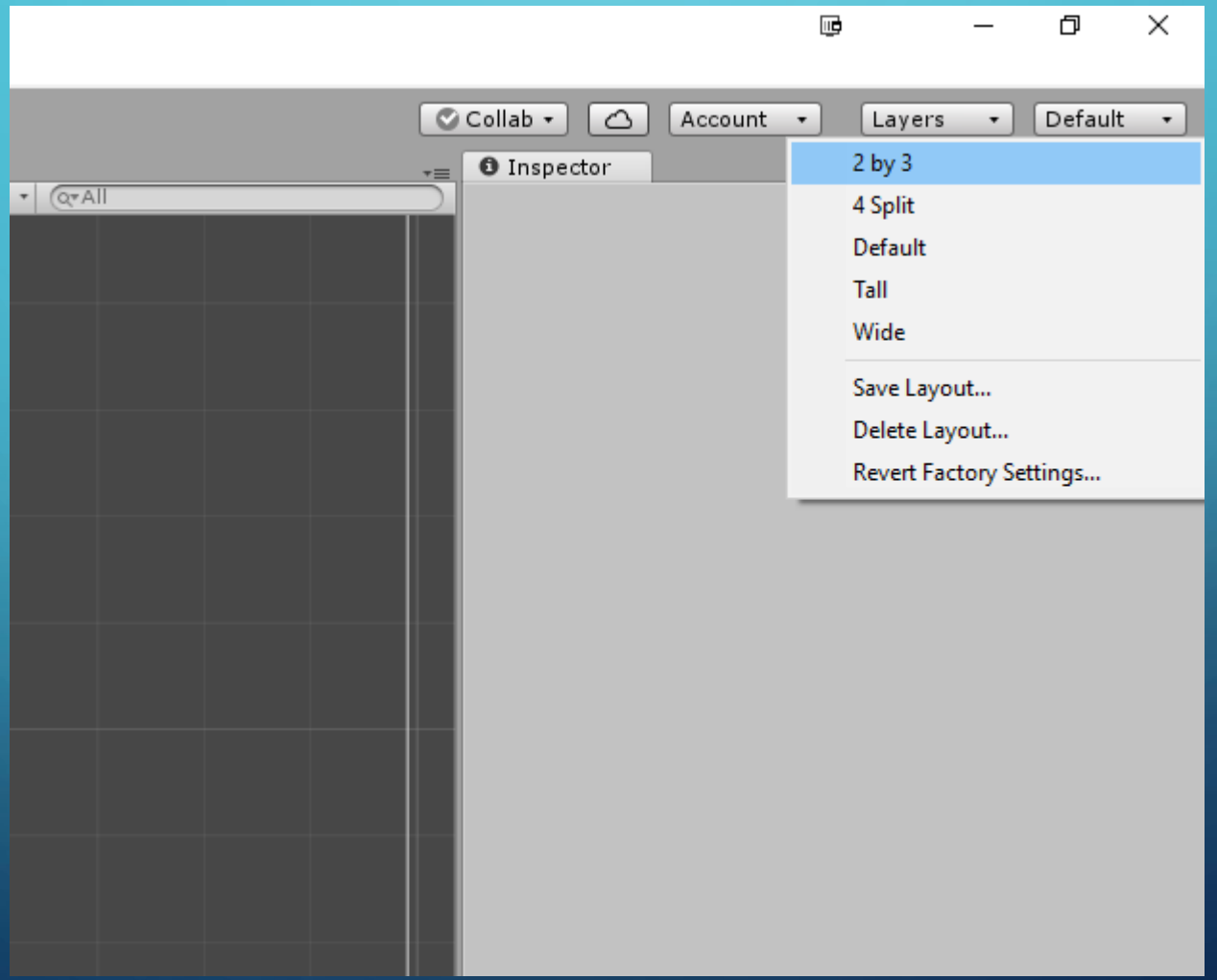
POBRANIE PROJKETU

- Projekt znajduję się na GitHub'ie: https://github.com/dawidraszka/FallingBall
- Pobrać zipa można za pomocą: Clone or download -> Download ZIP



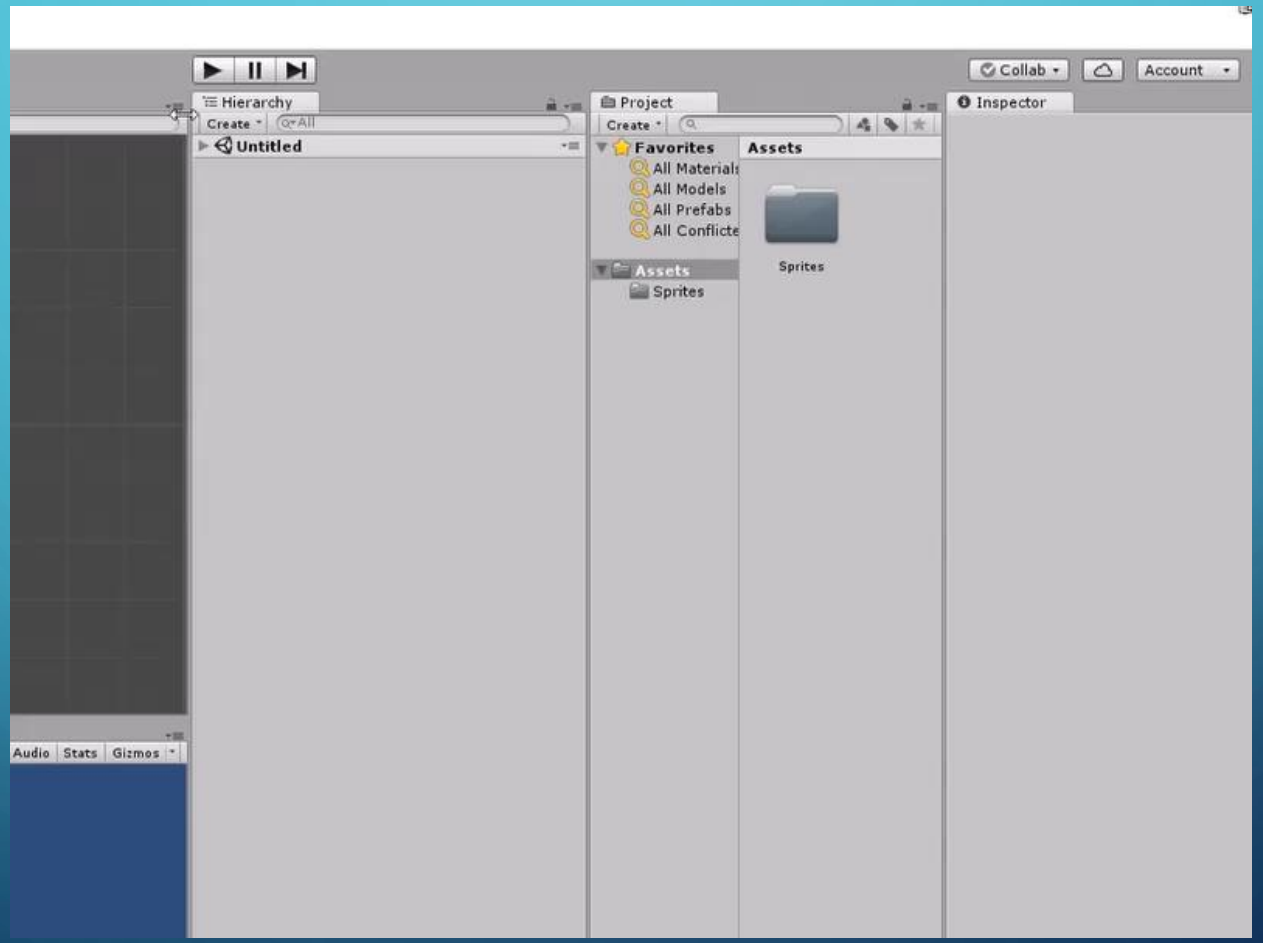
USTAWIANIE SCENY I PORUSZANIE SIĘ PO NIEJ

Ustawcie layout na „2 by 3”

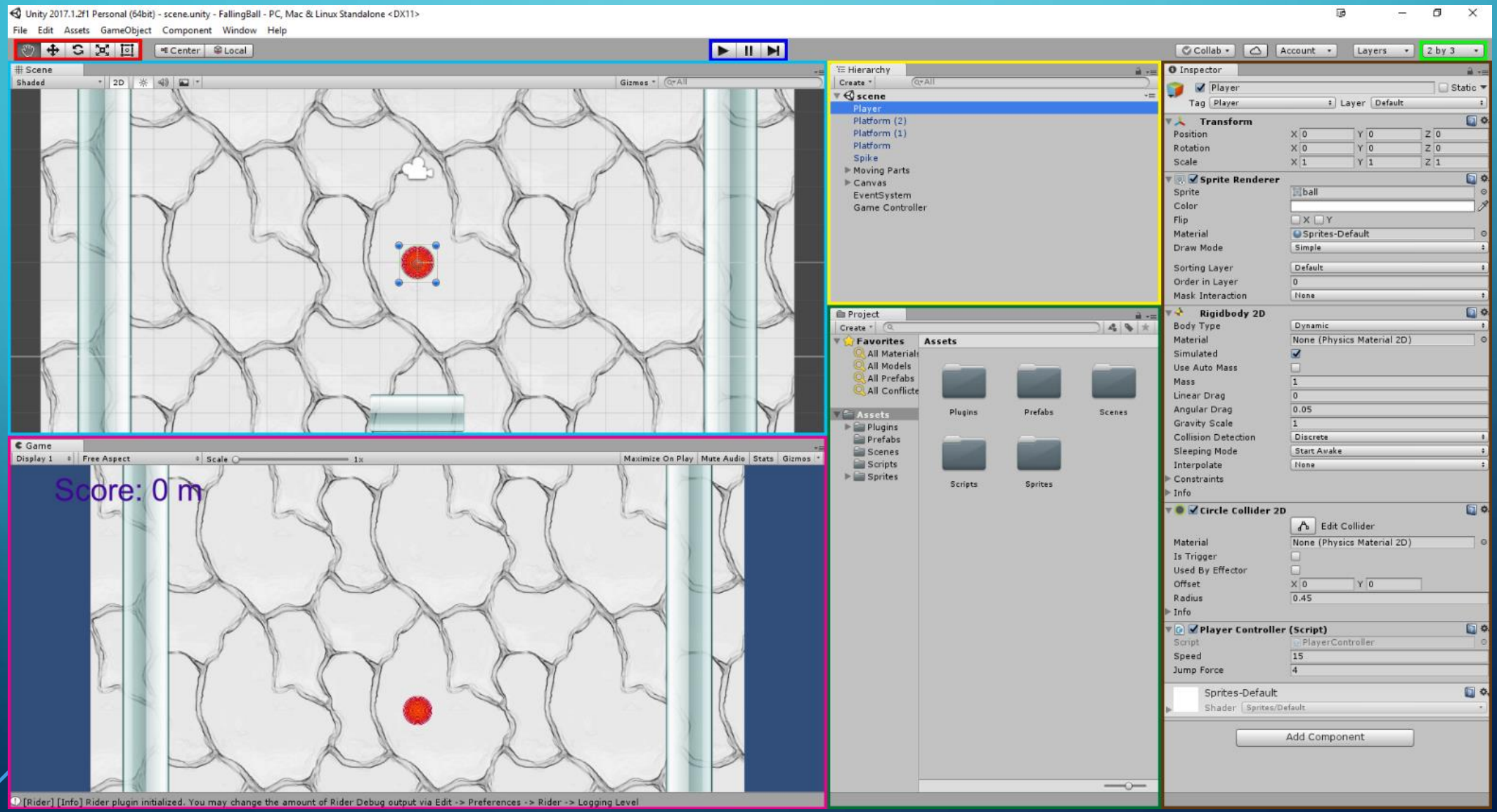


USTAWIANIE SCENY I PORUSZANIE SIĘ PO NIEJ

…A następnie przeciągnijcie okno „Hierarchy” nad „Project”



USTAWIANIE SCENY I PORUSZANIE SIĘ PO NIEJ



KOLIZJE ORAZ PODSTAWOWA FIZYKA

Fizyka

- Za umożliwienie obliczenia fizyki odpowiada komponent Rigidbody
- Zalecane jest dokonywanie obliczeń fizyki w funkcji FixedUpdate(), bo jest wywoływana zaraz przed każdym krokiem obliczeń fizyki przez silnik
- Zalecane jest, aby skala obiektu w grze była podobna do rzeczywistej – tj. jeśli samochód w rzeczywistości ma 4 metry długości to w grze powinien mieć 4 jednostki
- Nawet bez żadnego kodu na Rigidbody będą oddziaływały siły – np. grawitacji
- Istnieją 3 typy: Dynamic, Kinematic oraz Static
- Więcej tutaj i tutaj

KOLIZJE ORAZ PODSTAWOWA FIZYKA

Kolizje

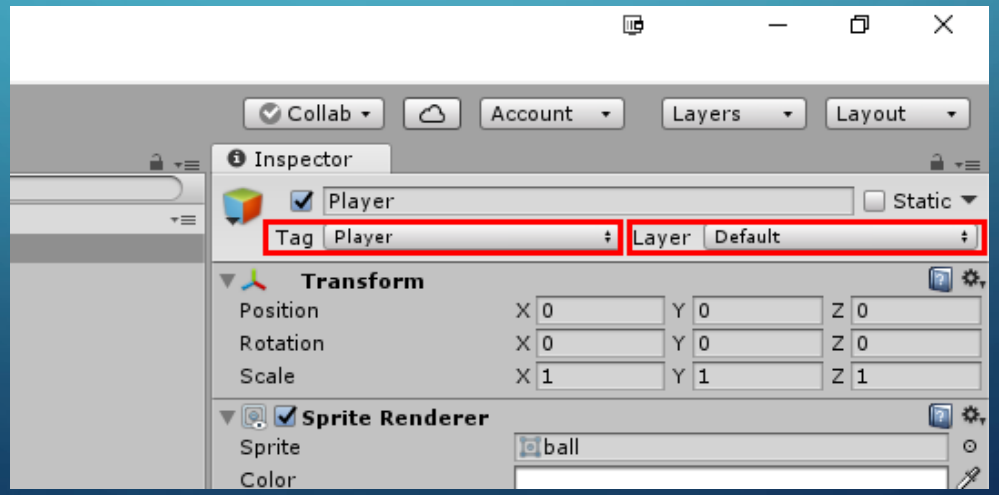
- Za obliczanie kolizji odpowiada komponent typu Collider
- Istnieją różne typy (np. Capsule Collider, Box Collider 2D czy
- Można ustawić jako Trigger (nie tworzy fizycznych kolizji, lecz je wykrywa)
- Nawet bez żadnego kodu na Rigidbody będą oddziaływały siły – np. grawitacji
- Aby kolizje działały poprawnie choć jeden z dwóch obiektów kolidujących musi mieć Rigidbody
- Więcej tutaj

DODAWANIE SKRYPTÓW, PODSTAWOWE FUNKCJE

- Podstawowe funkcje wywoływane w skryptach to: Start(), Update(), FixedUpdate()
- Start() jest wywoływane przy tworzeniu obiektu z podpiętym skryptem, używa się jej do inicjowania wartości. Więcej tutaj.
- Update() jest wywoływane co klatkę. Najczęściej używana funkcja do tworzenia mechanik gry. Więcej tutaj.
- Aby gra była niezależna od liczby klatek używa się Time.deltaTime. Więcej tutaj.
- FixedUpdate() jest wywoływane co stały czas i powinno być używane do obliczeń na Rigidbody. Więcej tutaj.

WARSTWY I TAGI

- Istnieją trzy rodzaje warstw i tagów: Tag, Sorting Layer oraz Layer.
- Tag służy do oznaczania konkretnych obiektów – np. Player
- Sorting Layer służy do układania obrazów 2D (podobnie do programów graficznych)
- Layer służy do oznaczania grup obiektów (np. „Ground”, które oznaczać będzie ziemię jak i platformy po których można skakać)
- Więcej tutaj.



PRZESZKODY W GRZE, PREFABY

- W tej grze przeszkody nie potrzebują Rigidbody – nie będą działały na nie żadne siły a kolizje będą wykrywane, bo już Player ma Rigidbody
- Polygon Collider 2D – typ collider'a, który pozwala na dowolną modyfikację jego kształtu. Obliczanie kolizji to skomplikowany proces, więc warto robić możliwie proste collider'y
- Prefabs – folder, w którym przechowuje się „szablony" obiektów. Z nich możemy tworzyć kilka instancji danego obiektu (np. przeciwnika czy przeszkody)
- Więcej o prefabach, tutaj.

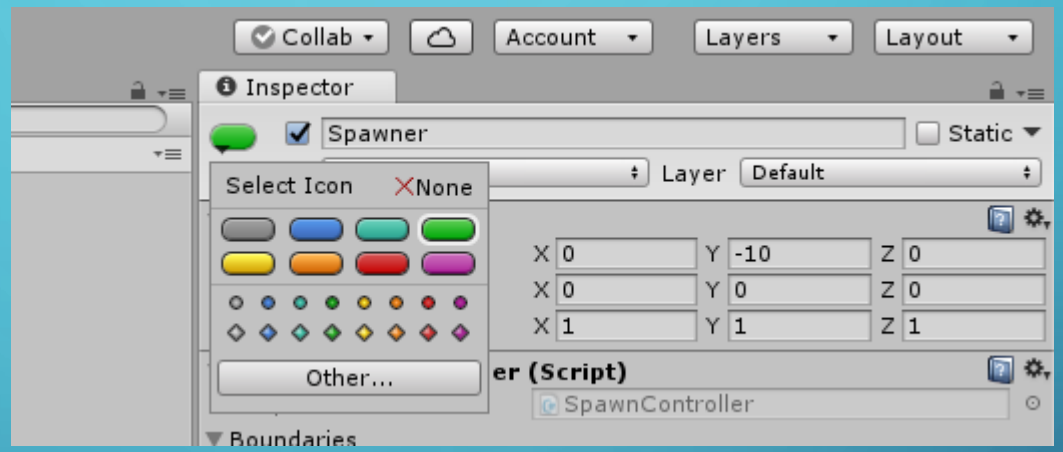
PORUSZANIE KAMERY

- Aby uczynić poruszanie kamery niezależnym od liczby klatek na sekundę, użyjemy Time.deltaTime
- Time.deltaTime jest to wartość ile czasu zajęła poprzednia klatka (w sekundach).
- Mnożenie przez tą wartość to jak powiedzieć „Chcę przesunąć ten obiekt o 5 metrów na sekundę” zamiast „Chcę przesunąć ten obiekt o 5 metrów na klatkę”
- Przykład:

```
void Update() {
        transform.Translate(0, 0, 5 * Time.deltaTime);
}
```

TWORZENIE OBIEKTÓW Z POZIOMU KODU

- Unity pozwala na dodanie ikon do obiektów – przydatne, gdy obiekt będzie w grze niewidzialny



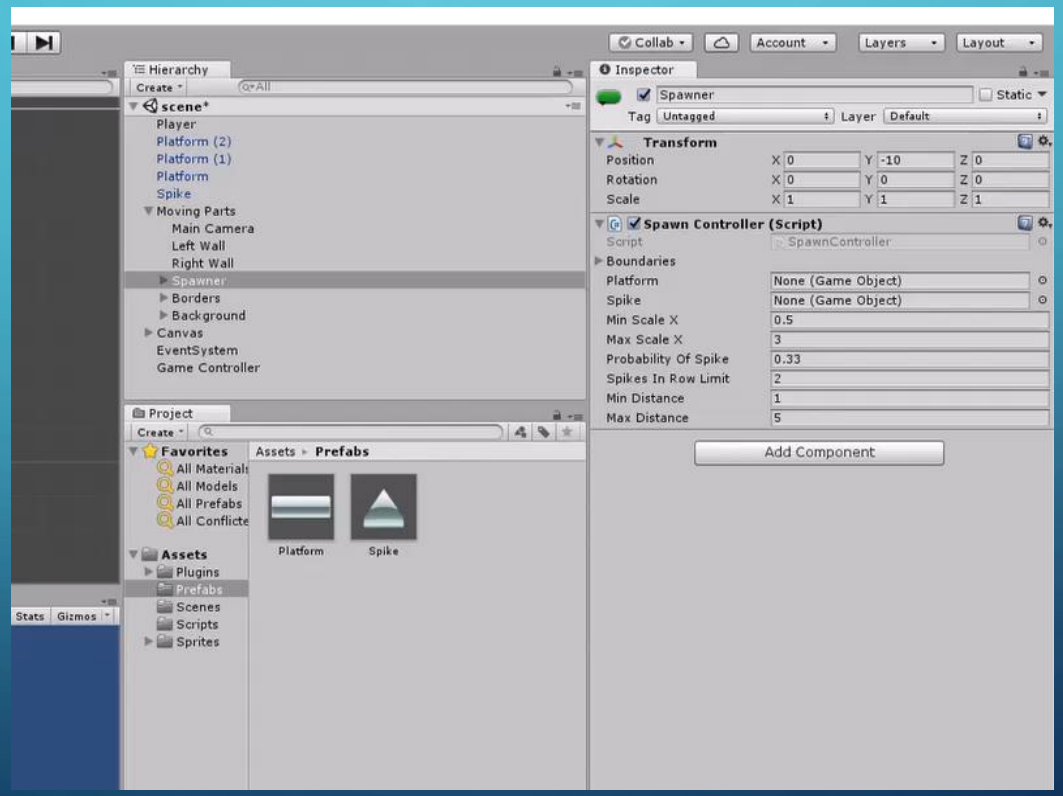
- Do tworzenia nowych obiektów używa się funkcji Instantiate(Object original) – tworzy ona kopię przekazanego Game Object'u
- Więcej o Intantitate tutaj.

TWORZENIE OBIEKTÓW Z POZIOMU KODU

- Game Object'y można przekazywać za pomocą publicznych zmiennych w kodzie np.:

```
public GameObject platform;
public GameObject spike;
```

- A następnie przeciągnąć odpowiedni obiekt z poziomu Unity

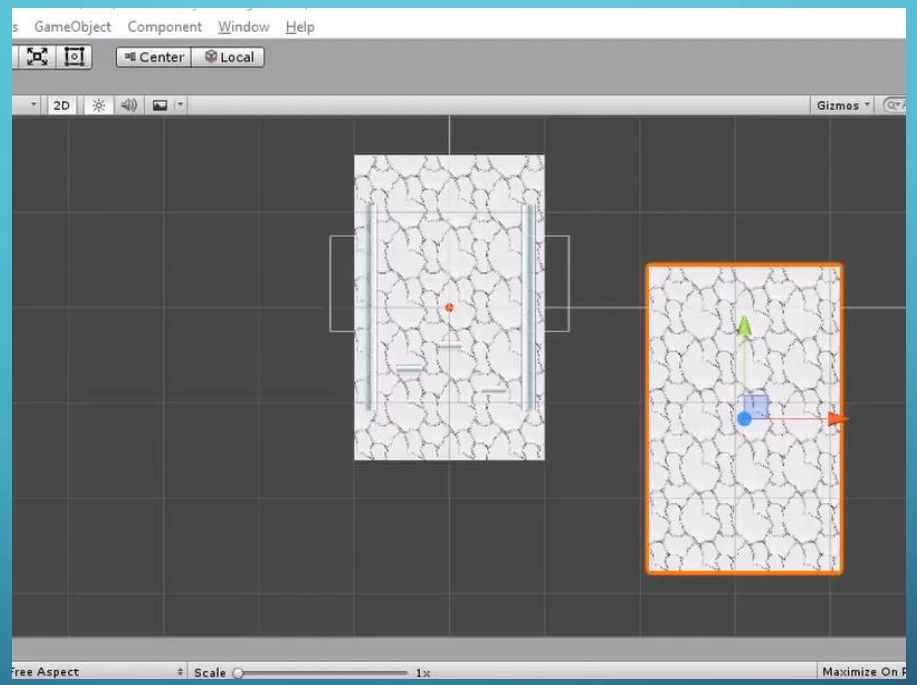


USUWANIE OBIEKTÓW Z POZIOMU KODU

- Z powodu oszczędności zasobów, niepotrzebne obiekty powinny być usuwane ze sceny
- W tej grze, obiekty pojawiają się w nieskończoność (lub do przegranej), więc trzeba je usuwać
- Kolizje „usuwaczy” służą tylko do wykrycia obiektu do zniszczenia – w związku z tym są w trybie Trigger
- Dolny border nie będzie miał Rigidbody, ponieważ reaguje tylko na gracza

PRZESUWAJĄCE SIĘ TŁO

- Aby tło było dość długie, tworzymy jego dwie kopie, jedna za drugą. Można tego dokonać przy pomocy klawisza V



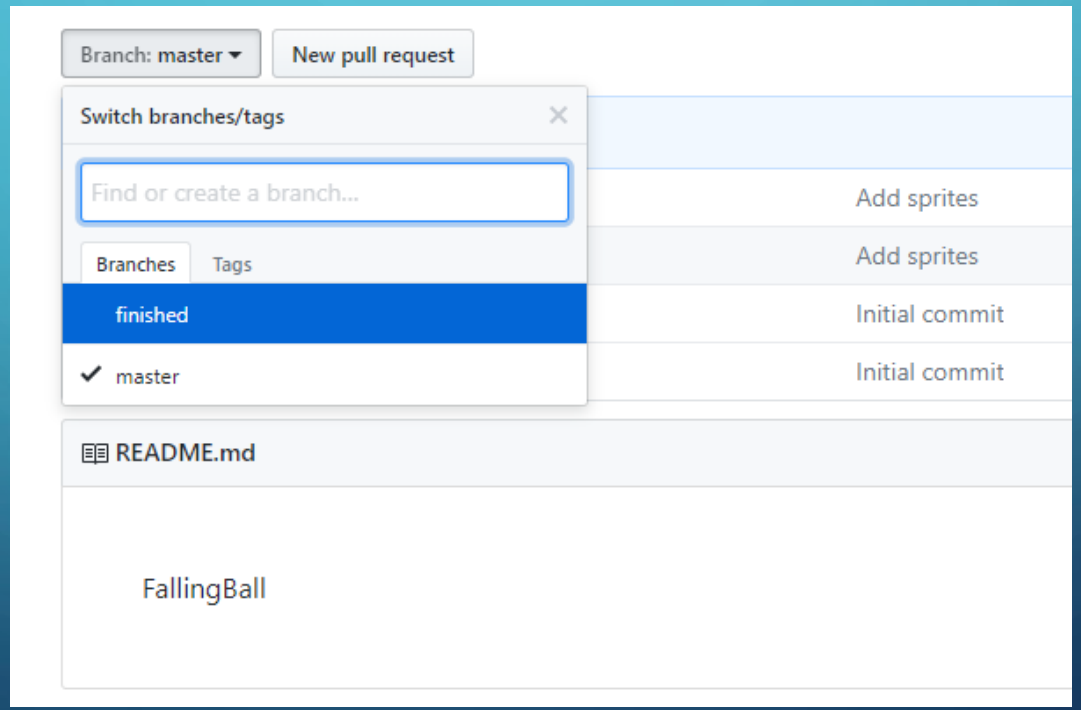
- Skrypt będzie przesuwał tło aż do wykonania pełnego przejścia (tzn. gdy każdy punkt tła przesunie się w górę o jego długość)

UI

- Każdy element UI umieszczany jest na Canvas'ie – jest to specjalny typ obiektu, który lepiej wspiera różne rozdzielczości niż zwykłe obiekty.
- Obiekty umieszczone w Canvas'ie ustawia się względem niego samego – a nie świata gry
- Ani Canvas ani obiekty w nim nie posiadają komponentu Transform
- Więcej tutaj.

UKOŃCZONY PROJEKT

- Znajduje się na GitHub'ie: https://github.com/dawidraszka/FallingBall
- Umożliwia prześledzenie kolejnych commitów
- Aby pobrać, należy przełączyć gałąź na finished



KONIEC

AUTOR: DAWID RASZKA