**Dolnośląska Szkoła Wyższa**

Yevhenii Kurbaiev

Nr. Indeksu 43298

**Gra platformowa**

**“Murmio 2D”**

Projekt specjalnościowy

Kierunek: Informatyka

Specjalność: Programowanie Gier Komputerowych

**Spis treści**

1. **Wstęp**
2. **Wybór tematu**
3. **Badanie rynku**
4. **Wybór technologji**
5. **Wymagania funkcjonalne i niefunkcjonalne**
6. **Koncepcja gry**
7. **Projekt graficzny**
8. **Omówienie kodu**
9. **Testy wydajności**
10. **Wstęp**

Branża gier kompyterowych na dzień dzisiejszy jest jedną z największych gałęzi mediaproduktów. Prawdopodobnie, że gra komputerowa Grand Theft Auto V jest najbardziej dochodowy produkt medialny. Według informacji MarketWatch przez 4 lata Take- Two(wydawca gty) zarobila w okolicy 6 milliardów dolarów amerykańskich, sprzedając 90 millionów kopji gry. Też GTA V do 2017 była pierwsza w topach sprzedaż w Europie i Wielkiej Brytanii. Młodzież też prefetuję grać w gry komputerowe. Co powoduje, że w tej branże „kręce się” duża ilość pieniędzy. Też przewagą gier komputerowych nad „konkurentami” jest obecność interaktywności, co pomaga graczu bardziej asocjować siebie z przgodą na ekranie.

1. **Wybór tematu**

Wśrod developerów gier Indie są popularne gry w formacie 2D. Indie-developery preferują ten typ gier, bo są one łatwiejsze do programowania w poruwnaniu do gier 3D. Też gry 2D mniej obciążają komputer, dlatego większy zasięg rynkowy. Jednym z najpopularniejszych gatunków gier jest Platformer(lub Gra Platformowa). Platformery zostały wymyślone w latach 80 ubiegłego wieku, kiedy komputery i konsoli nie były potężne, i dalej są popularne. Dlatego wybrałem Grę Platformową

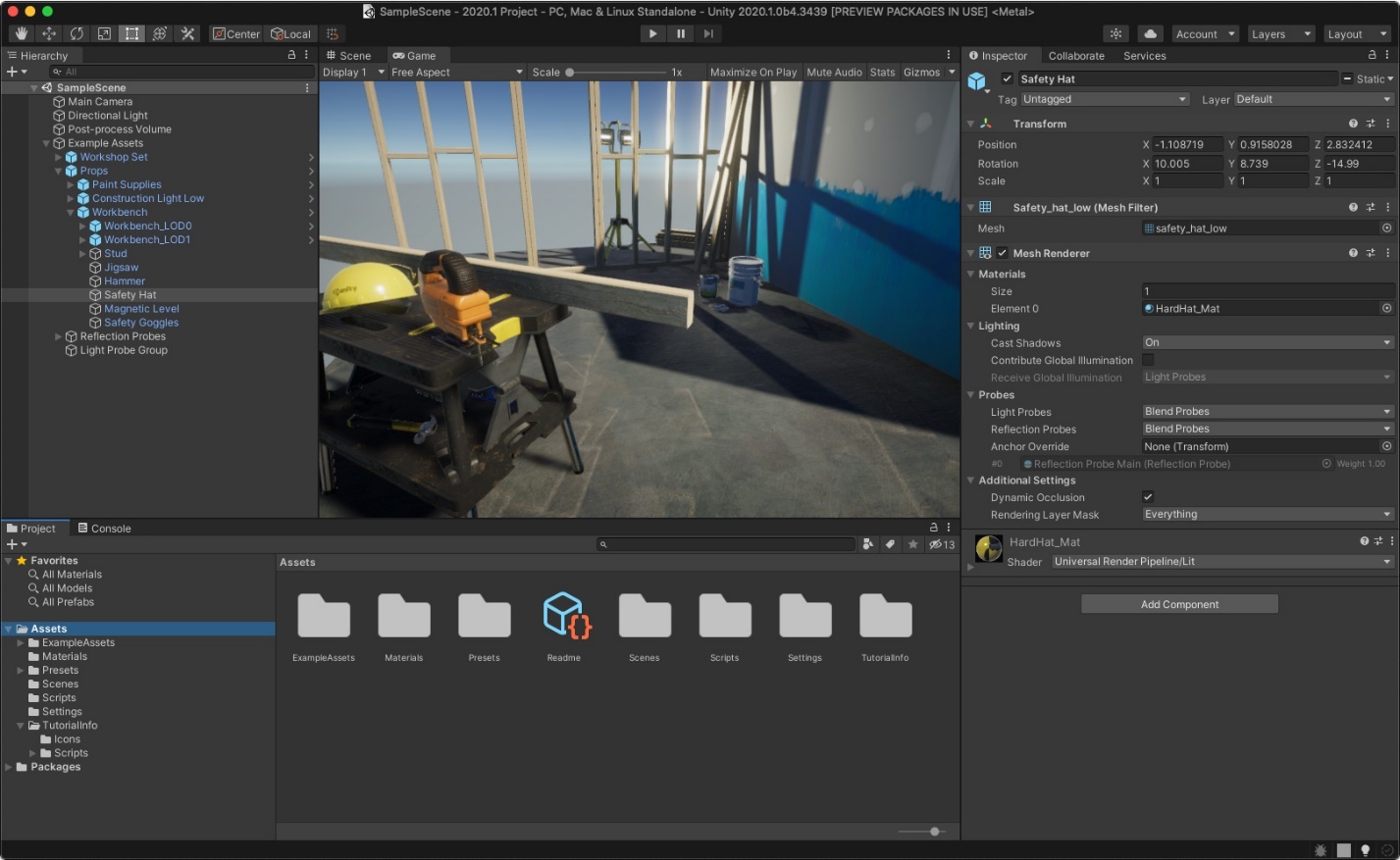
1. **Badanie rynku**

Popularnych platformerów jest dużo, ale najbardziej popurlarną serią platformerów jest seria gier pro Mario. W marcu 2022 roku seria Super Mario sprzedała się w ponad 396,80 milionach egzemplarzy na całym świecie i zarobiła ponad 22 miliardy dolarów w szacowanych przychodach ze sprzedaży. W marcu 2022 roku cała seria Mario sprzedała się na całym świecie w ponad 776,80 egzemplarzach, co czyni ją najlepiej sprzedającą się serią gier wideo wszech czasów. „Pierwszy” Mario powstał w 1985 roku dla konsoli NES i nazywał się „*Super Mario Bros”*. Był sprzedany w 40,240,000 egzemplarzach i osiągnął zysk w $1,584,000,000.

1. **Wybór technologji**

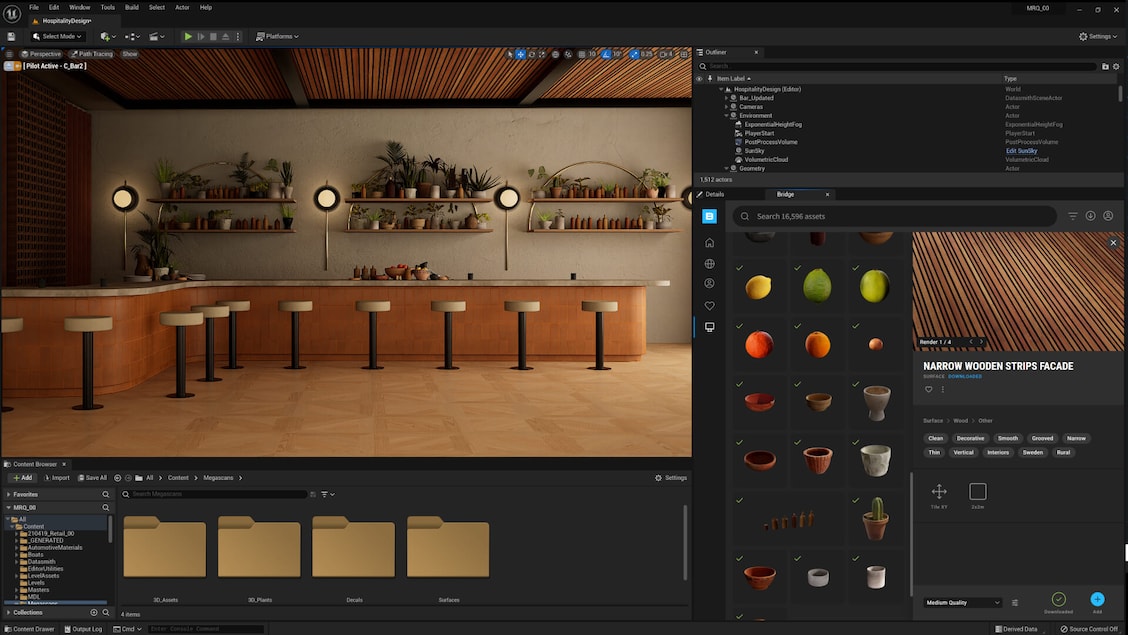
Dla napisania gry, podobnej do *Super Mario Bros* mi jest potrzebny systyem, który by sterował raycastingiem, systemem kolizji a inputem. Dla tych potrzeb będzie optymalnie wybrać jeden z dostępnych darmowych silników gier:

**Unity Engine:**



[*https://unity3d.com/beta/2020.1b*](https://unity3d.com/beta/2020.1b)

**Unreal Engine:**



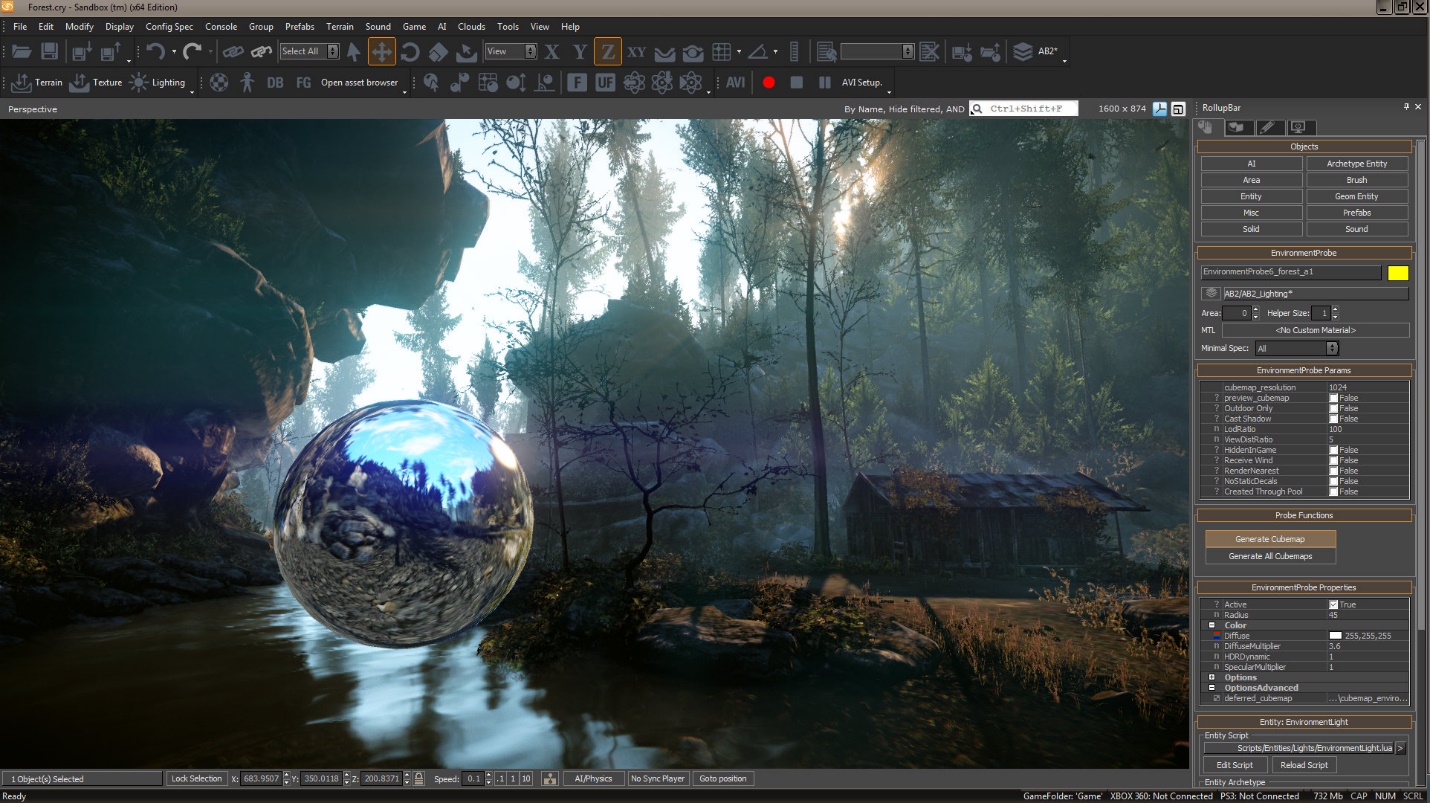
[*https://www.unrealengine.com/en-US/unreal-engine-*](https://www.unrealengine.com/en-US/unreal-engine-)*5*

**Cocos Engine:**



[*https://www.cocos.com/en/creator*](https://www.cocos.com/en/creator)

**Cry Engine:**



[*https://www.videotesty.pl/news/3784/silnik-cryengine-na-dwoch-zwiastunach/*](https://www.videotesty.pl/news/3784/silnik-cryengine-na-dwoch-zwiastunach/)

Mój wybór padł na Unity Engine. Ten śilnik ma dużą społeczność, co oznacza, że w razie problemów będę mógł szybciej ich załatwic. Również moin zdaniem documentacja Unity jest bardziej wygodna w porównywaniu do konkórentów. Też Unity oferuje darmową wersje PRO dla studentów.

1. **Wymagania funkcjonalne i niefunkcjonalne**

Wymagania funkcjonalne:

- skok na platformy

- zbieranie bonusów

- skok na wroga (z zamiarem zabicia go)

- omijanie wrogów

Wymagania niefunkcjonalne:

- stabilna praca na platformie docelowej: Windows 10 64 bit.

- gra powinna być płynna (przynajmniej 30 FPS)

1. **Koncepcja gry**

Koncepcja gry jest podobna do Super Mario Brose:

gracz musi ukończyć poziomy i nie umrzeć. Gracz może przeskakiwać na platwofmy lub inne objekty w grze z celem omijania wrogów, lub grać może przeskakiwać nad wrogami, lub skakać na wrogów z celem zabicia wroga. Jeżeli wróg „złapi” gracza – gra jest skończona z porażką.

Jeżeli grać nie umrze w trakcie gry i dojdzie do końca – gra jest skończona z sukcesem

1. **Projekt graficzny**

Assety dla mojej gry ja wziąłem z tej strony:

<https://weeklycoder.com/download/assets-for-mario-clone-tutorial/>

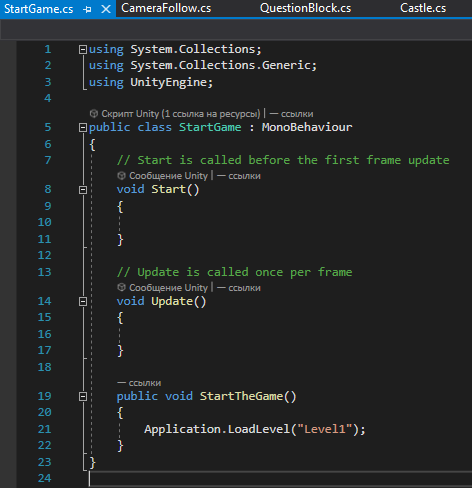
te assety już zawierały sprite’y, animacji i tilemap’y. Również są to assety z liberalną licencją

1. **Omówienie kodu:**

Przy projektowaniu gry korzystałem z koncepcji programowania komponentnego (Entity Component System).

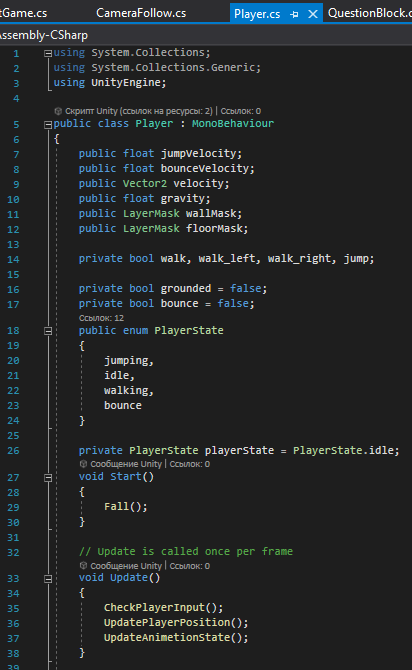
Zaletą tego wzożca jest bezpieczeństwo gry od przypadkowego błedu, żeby inny component sie zie zepsuł

Pierwszy skrypt, który zacznie działać po uruchomienu gry nazywa sie StartGame.cs

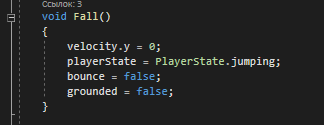


Służy on do tego, żeby po kliknięciu na knopkę „Play” wywołać funkcję StartThe Game, która załaduje pierwszy poziom

Skrypt Player.cs opisuje zachowanie gracza

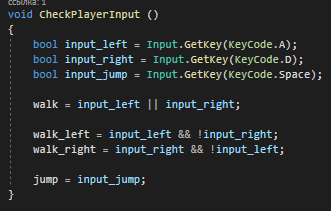


Zmienne publiczne są po to, żeby edytować ich wartość na poziomie interface’u silnika. Funkcja Start() odbywa sie tylko raz, przy uruchomieniu. W tej funkcji wywołam funkcje Fall(), żeby grać zaczął padać, dopóki nie spadnie na ziemie.

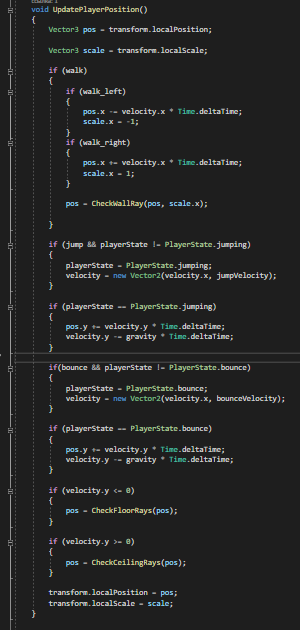


Funkcja Update() jest wywoływana każdy frame.

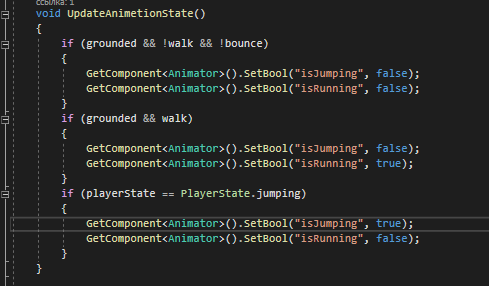
Najpierw ja sprawdzam inputy, którę wprowadził grać za pomocą funkcji CheckPlayerInput()



Dalej przesuwam gracza, jeżeli jakaś akcja sie zdażyła(grać nacisnął A/D/Spacje) za pomocą funkcji UpdatePlayerPosition();



Dalej zmieniam animacje, jeżeli stan gracza sie zmienił za pomocą funkcji UpdateAnimetionState();



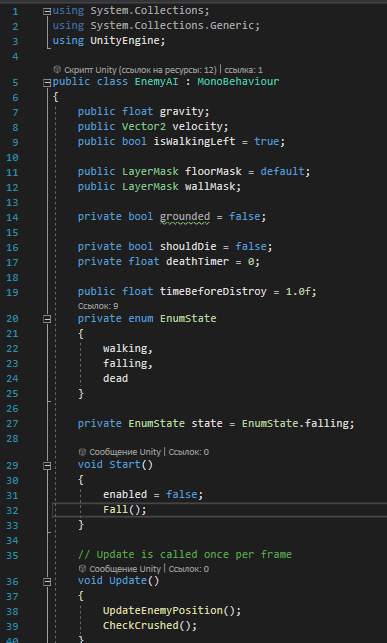
W funkcjach CheckCeilingRays(Vector3 pos) a

Vector3 CheckFloorRays(Vector3 pos)

Za pomocą metody Raycasting’u sprawdzam, czy nie natknął się

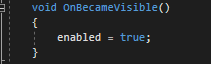
Grać na scianę, czy stoi na podłodze lub czy nie spotkał wroga. W zależności od sytuacji podejmuje różne dycyzji.

Script EnemiAI służy po to, żeby opisać zachowanie postaci-wroga

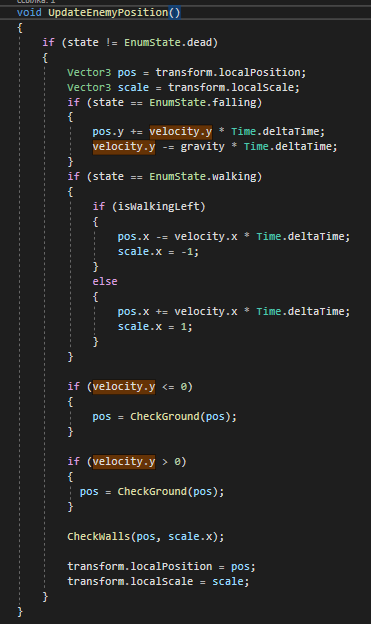


Na starcie gry ja wyłączam Script do momenty, kiedy gracz „zobaczy” wroga. Włączam go za pomocą funkcji

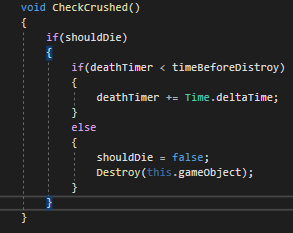
OnBecameVisible()



Za pomocą funkcji UpdateEnemyPosition(), jeżeli postać zyje, rucham ją

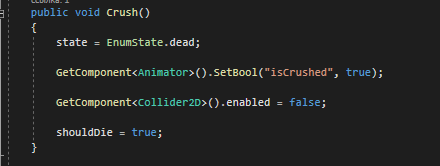


Funkcja CheckCrushed() służy po to, żeby zniszczyć objekt Enemy za sekundę po zabiczu postaci graczem, żeby zaoszczędzić pamięć

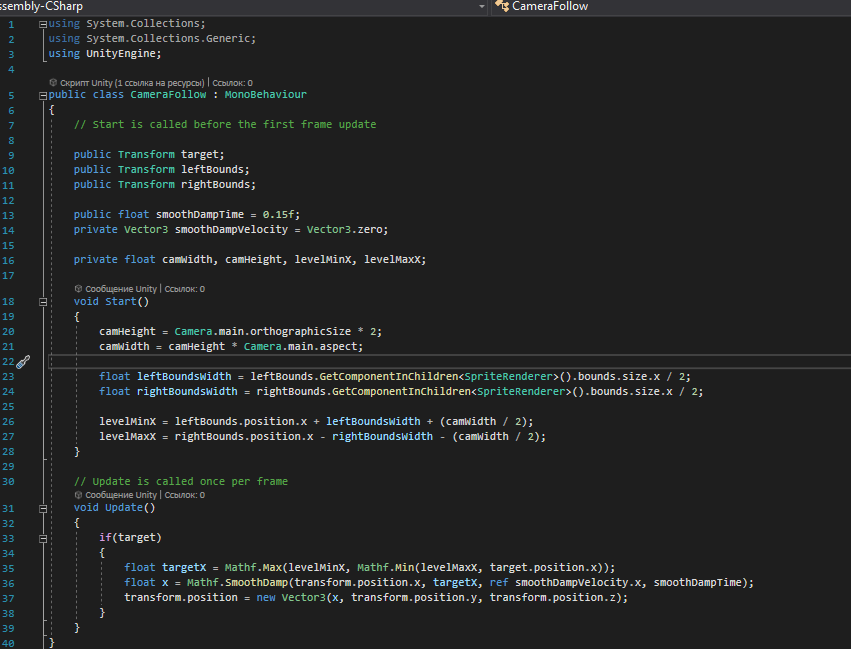


Publiczna funkcja Crush() jest wywoływana w skrypcie Player.cs,

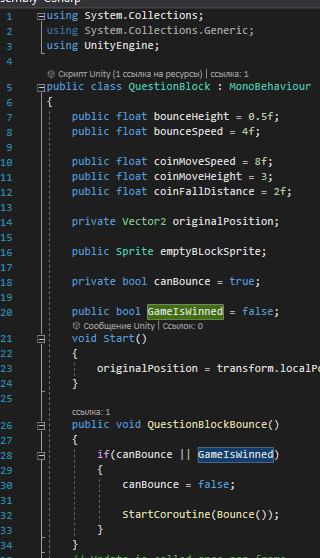
Ona nadaje status postaci jako ‘dead’, wyłącza Animator i Collider2D.



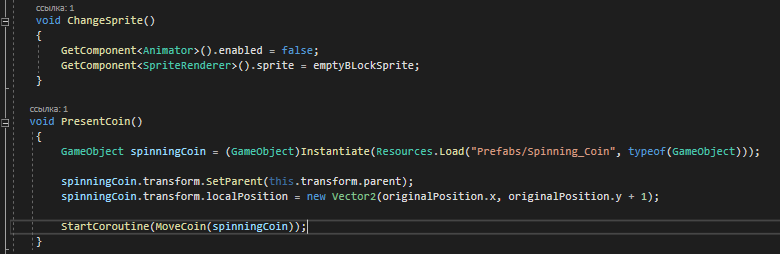
Script CammeraFollow jest podpięty pod kamerę, i służy, żeby kamera śliedziła za graczem, płynie poruszając i nie „wychodząc” za granicę pozioma



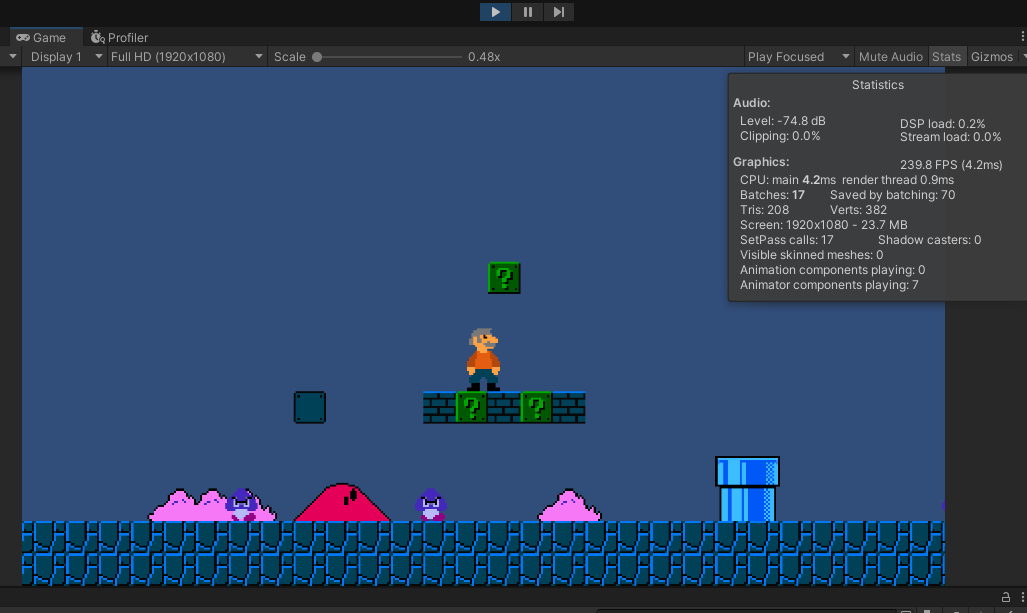
Script QuestionBlock.cs służy żeby przełączać tryby bloka ze znakiem zapytania.



Publiczna funkcja QuestionBlockBounce jest wywaływana w scripcie Player.cs. Ona zaniemia sprite na emptySprite, podnosi block i generuje monetę, która znika za jakiś czas



1. **Testy wydajności**



Gra pukazuje stabilne 200+ fps, co spełnia wymagania niefunkcjonalne. Testy były przeprowadzone przy konfiguracji:

Processor: Intel Core i5-8250u

RAM: 12 gb 2133 taktowanie

Karta graficzna: Nvidia MX150 2gb

Link na kod źródłowy:

<https://github.com/sotryhardy/Murmio-2D>

Link na gotowy build:

<https://m365ht-my.sharepoint.com/:f:/g/personal/dsw43298_student_dsw_edu_pl/EsWRb74DuetHh5PrQgSdMKoBXGdzJsqwHnzb0OHBQzmSvA?e=mxYIk0>

Bibliografia

<https://www.marketwatch.com/story/this-violent-videogame-has-made-more-money-than-any-movie-ever-2018-04-06>

<https://vgsales.fandom.com/wiki/Mario#:~:text=As%20of%20March%202022%2C%20the,game%20franchise%20of%20all%20time>.

<https://weeklycoder.com/download/assets-for-mario-clone-tutorial/>