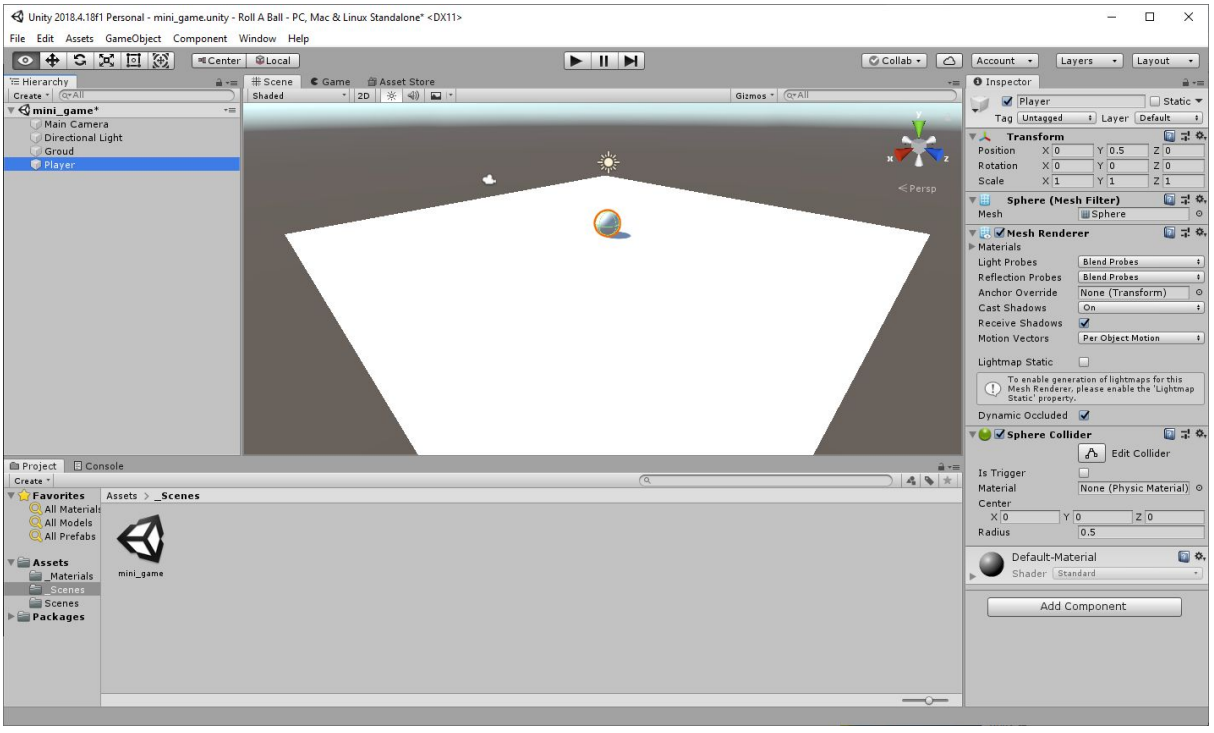
1. Cel zadania oraz wariant:

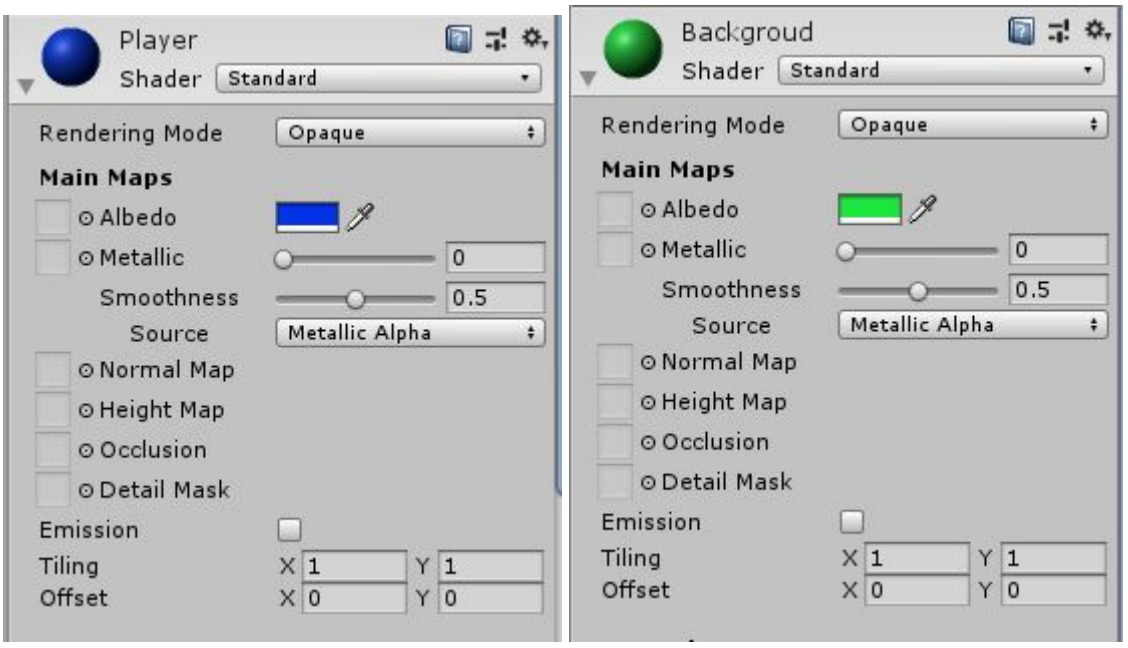
Opracować grę typu Roll a Ball z takimi parametrami
Wariant 1. Kolor materialu „playera” – niebieski,
obiekty „pick up” dwóch typów
1 typ obiekta „pick up” - cylinder,
2 typ obiekta „pick up” - capsule,
ilość obiektów „pick up” typu 1 - 9,
ilość obiektów „pick up” typu 2 - 3,
Kolor materialu obiektów „pick up” typu 1 – żółty,
Kolor materialu obiektów „pick up” typu 2 – czerwony,
Kolor materialu „ścian” - zielony
Reguły gry:
trafianie w obiekt typu 1 – 1 punkt
trafianie w obiekt typu 2 – 3 punkty
warunek zakończenia gry – 9 punktów

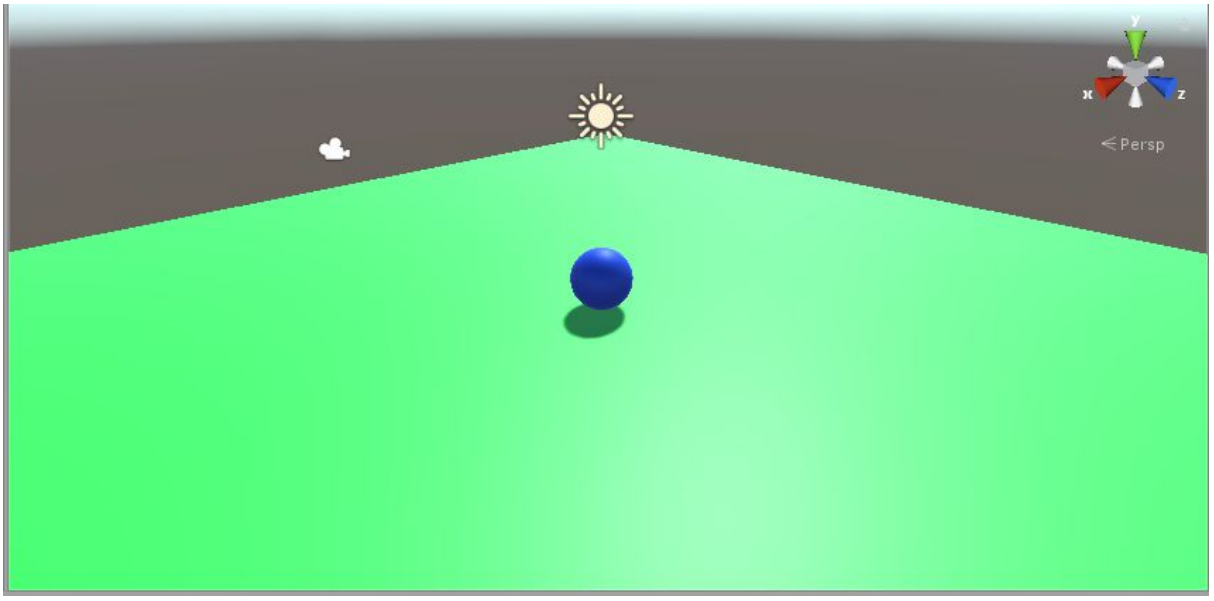
2. Przebieg ćwiczenia

Na początek należy utworzyć pole gry oraz obiekt gracza (puste pole oraz kule, którą będzie poruszać użytkownik). W tym celu wykorzystujemy obiekty Unity.
Z menu “GameObject” wybieramy opcje “3D Object”. Następnie tworzymy jeden element Plane (nazywamy go “Ground”) oraz obiekt gracza czyli Sphere (nazywamy go “Player”). Na zrzucie poniżej widzimy aktualną scenę.



W celu zmiany koloru kuli na niebieski (zgodnie z wariantem zadania), dodajemy folder “_Materials” i za pomocą odpowiedniego materiału, zmieniamy kolor gracza.
Zmiana koloru ma miejsce poprzez przesunięcie danego materiału na obiekt.
Na poniższych zrzutach można zobaczyć ustawienia koloru oraz widok aktualnego stanu gry.





Kolejnym krokiem jest utworzenie skryptu odpowiedzialnego za ruch gracza. W tym celu tworzymy folder “_Scripts”. Następnie wybierając obiekt “Player”, dodajemy komponent (“Add compenent”) “C# Script”.

Skrypt składa się z kilku ważnych rzeczy:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    public float speed;
    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }



    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

        rb.AddForce(movement * speed) ;

    }
}
```

Podstawowe elementy:

a) Speed (Prędkość) może zostać określona w programie Unity.
b) Rigidbody (ciało, w tym przypadku kula czyli “Player”)
c) Funkcja Start której zadaniem jest pobranie elementu
d) Funkcja FixedUpdate której zadaniem jest aktualizacja aktualnej pozycji kuli

Przy aktualnym stanie gry, możemy poruszać “Playerem” za pomocą strzałek oraz możemy zwiększać lub zmniejszać prędkość poruszania.

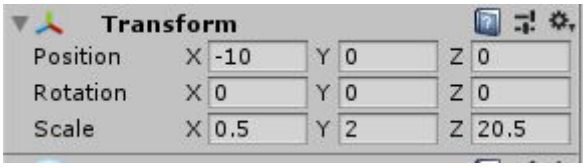
Następnym krokiem jest utworzenie skryptu, który będzie poruszał kamerą.

```
public class CameraController : MonoBehaviour
{
    public GameObject Player;
    private Vector3 offset;

    void Start()
    {
        offset = transform.position - Player.transform.position;
    }

    // Update is called once per frame
    void LateUpdate()
    {
        transform.position = Player.transform.position + offset;
    }

}
```

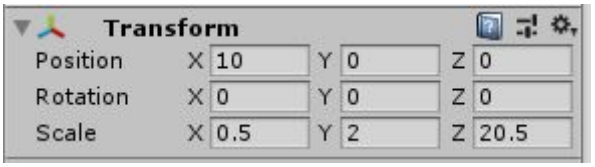
Ważnym elementem tego skryptu jest obiekt “Player”, który musimy przekazać w programie Unity.

Następnie tworzymy ściany ograniczające pole gry:
Używamy do tego “Unity Cube” (tworzymy element 3D Cube).
Współrzędne dla ścian:

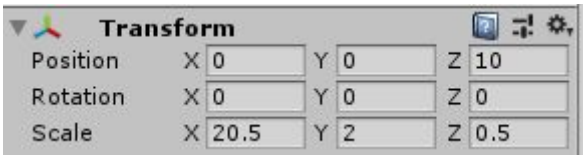
a) Zachodnia



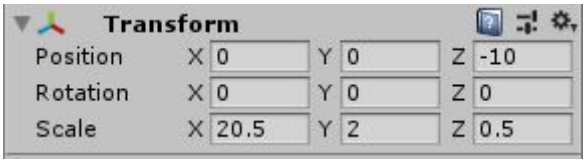
b) Wschodnia



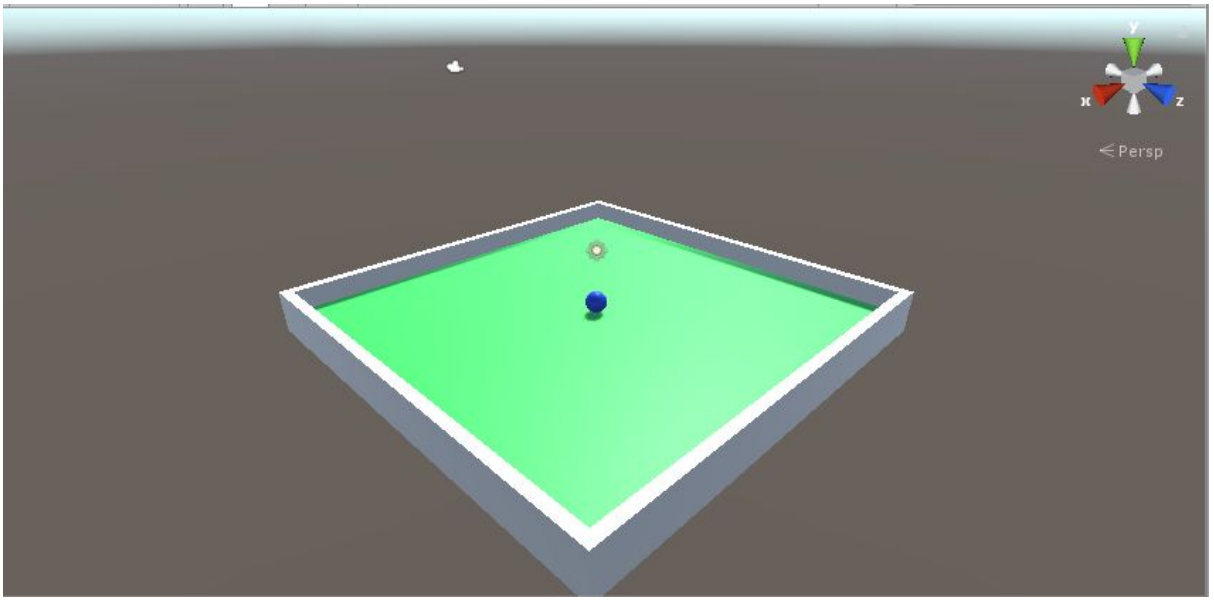
c) Północna



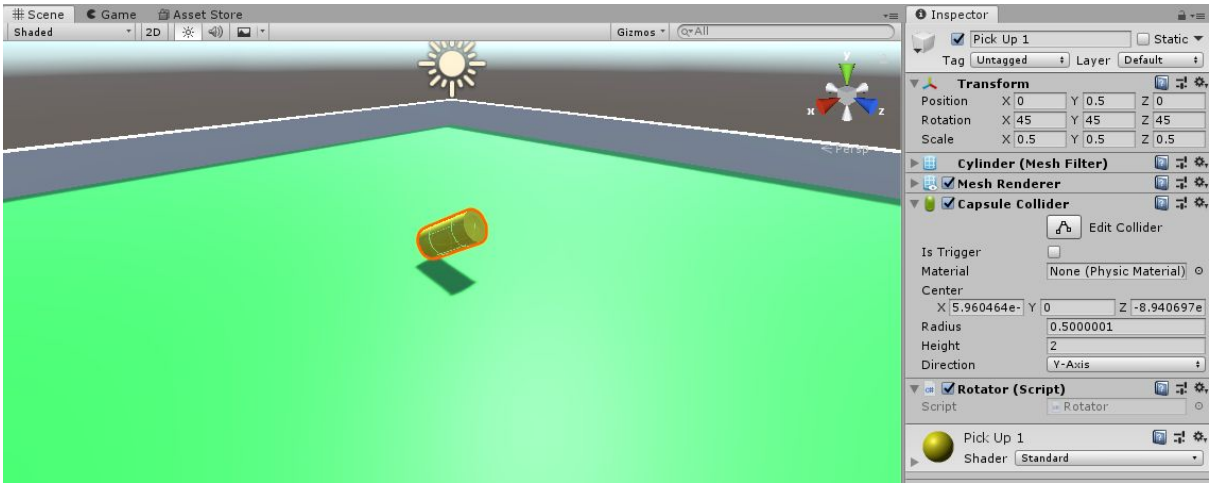
d) Południowa

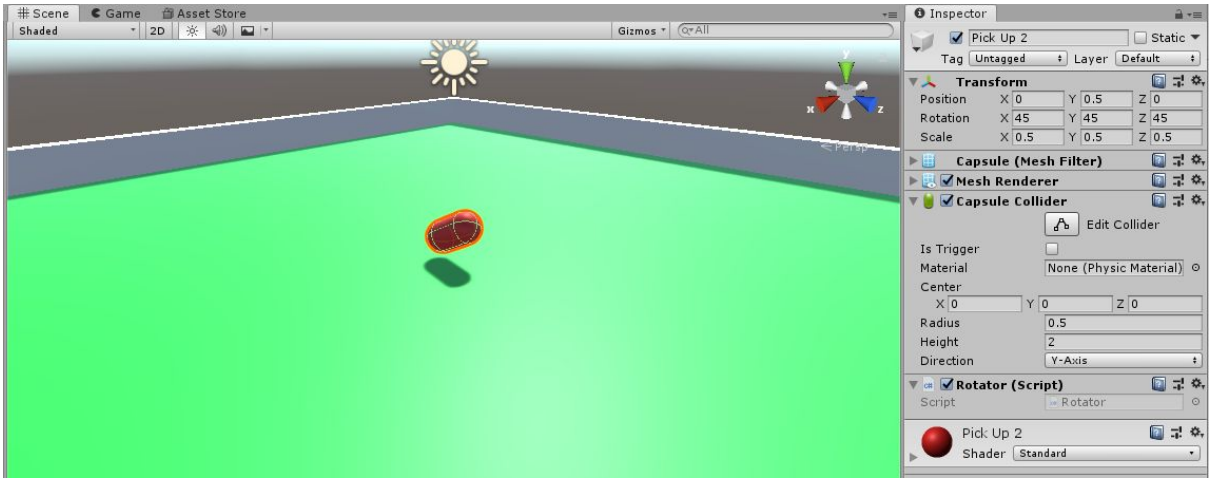


Efekt końcowy:

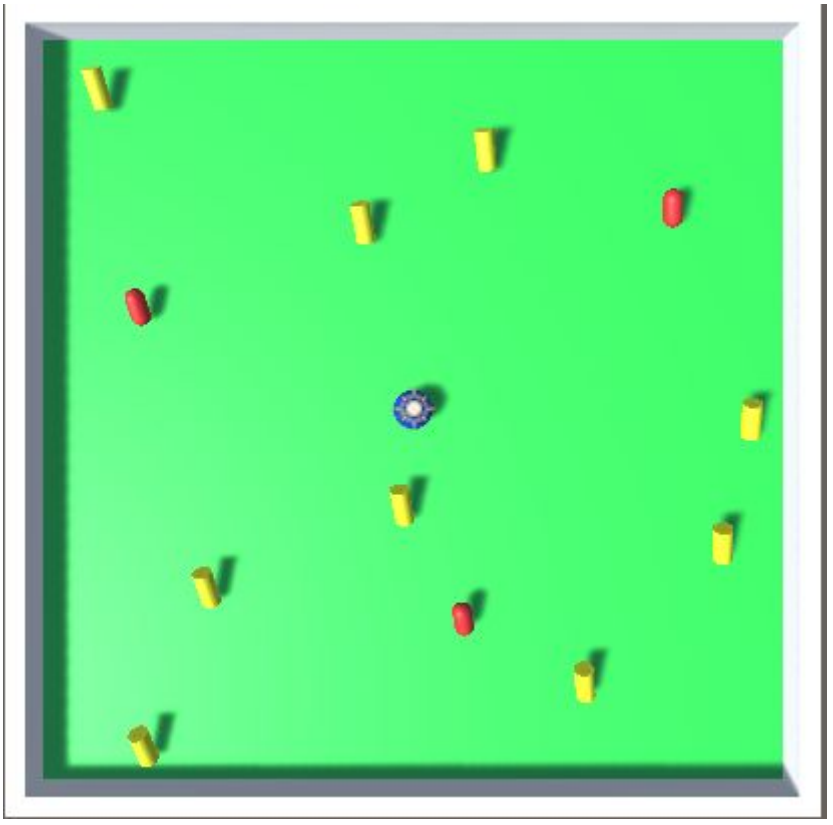


Następny krok dotyczy tworzenia elementów “Pick Up”. Tworzymy nowy materiał, który przypisujemy do nowo powstałych elementów.





Duplikujemy elementy zgodnie z wariantem



Ważne jest aby każdy element posiadał własne ciało oraz używał grawitacji i kinematyki.
Skrypt do “pochłaniania” elementów “Pick Up”:

```
void OnTriggerEnter(Collider other)
{
    if(other.gameObject.CompareTag("Pick Up 1"))
    {
        other.gameObject.SetActive(false);
    }
    if (other.gameObject.CompareTag("Pick Up 2"))
    {
        other.gameObject.SetActive(false);
    }
}
```

Po dodaniu zliczania punktów kolizja wygląda następująco:

```
void OnTriggerEnter(Collider other)
{
    if(other.gameObject.CompareTag("Pick Up 1"))
    {
        other.gameObject.SetActive(false);
        score += 9;
        scoreText.text = "Aktualny wynik: " + score.ToString();
    }
    if (other.gameObject.CompareTag("Pick Up 2"))
    {
        other.gameObject.SetActive(false);
        score += 3;
        SetCountText();
    }
}
```

Natomiast obiekt “Player” posiada już dwie zmienne pobierane z Unity

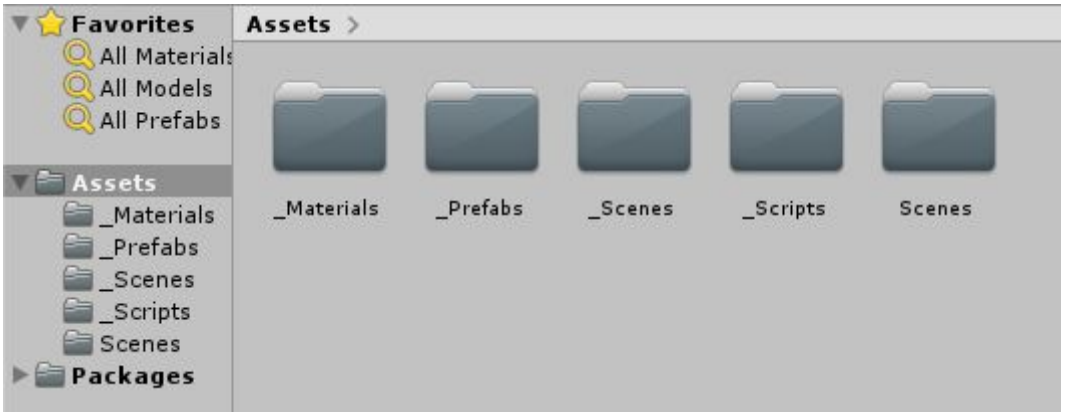
Następnie tworzymy warunek wygranej (winText).
Końcowy kod “Playera”.

```
{
    public float speed;
    private Rigidbody rb;
    private int score;
    public Text scoreText;
    public Text winText;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
        score = 0;
        SetCountText();
        winText.text = "";
    }



    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

        rb.AddForce(movement * speed) ;

    }

    void OnTriggerEnter(Collider other)
    {
       if(other.gameObject.CompareTag("Pick Up 1"))
        {
            other.gameObject.SetActive(false);
            score += 1;
            scoreText.text = "Aktualny wynik: " + score.ToString();
        }
        if (other.gameObject.CompareTag("Pick Up 2"))
        {
            other.gameObject.SetActive(false);
            score += 3;
            SetCountText();
        }
    }

    void SetCountText()
    {
        scoreText.text = "Aktualny wynik: " + score.ToString();
        if (score >= 9)
        {
            winText.text = "Wygrales!";
        }
    }
}
```

Końcowe przypisania dla “Playera”

Zrzut ekrany po osiągnięciu odpowiedniego wyniku

Solucja projektu:



3. Wnioski:

Unity posiada przyjazny dla użytkownika interfejs. Dzięki licznym “tutorialom” nauka w Unity staje się dużo prostsza. Środowiska tego można używać w celu tworzenia bardziej zaawansowanych gier. Przez ciągłe wsparcie twórców, powstają coraz to nowsze wersje oprogramowania, posiadające coraz większe możliwości.

Powyższe zadanie wymagało wykorzystania podstawowych elementów Unity, takich jak proste skrypty, tworzenie prostych obiektów. Zadanie nie było trudne do wykonania. Efekty działania programu zostały przedstawione na zrzutach ekranu w niniejszym sprawozdaniu.