

Laboratorium ZTPGK

Temat: Fizyka w grach komputerowych

Prowadzący: dr inż. Michał Staniszewski

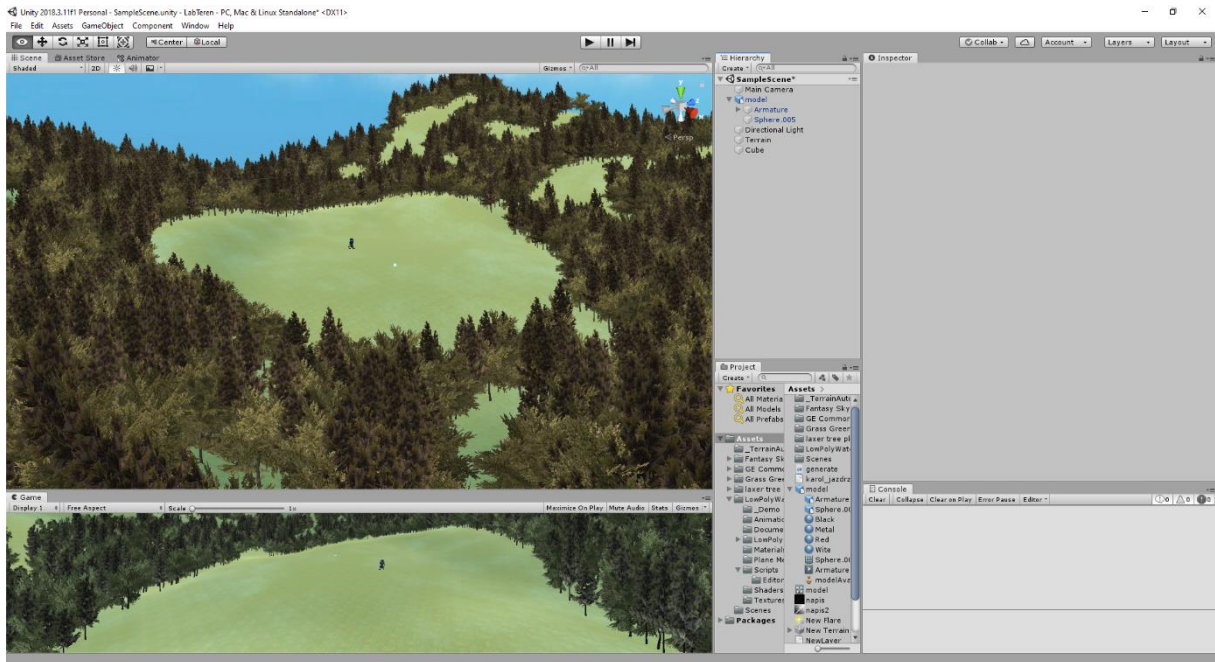
Data przeprowadzenia laboratorium: 20.03.2020 r.

IGT Inf sem1

Karol Jażdżyk

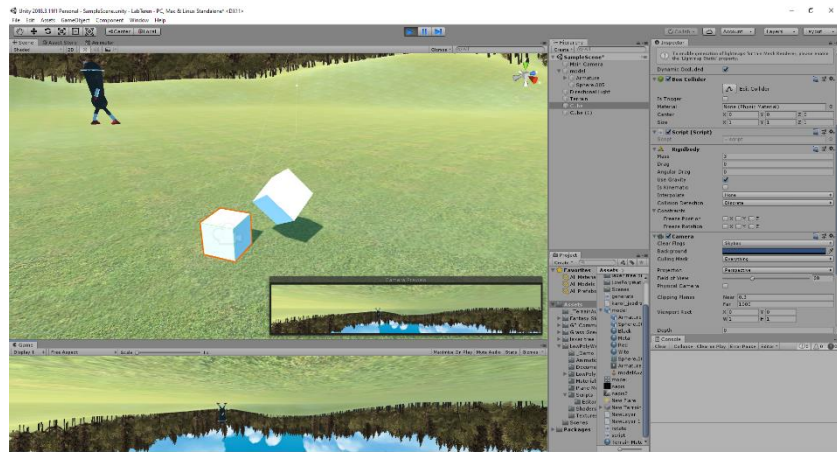
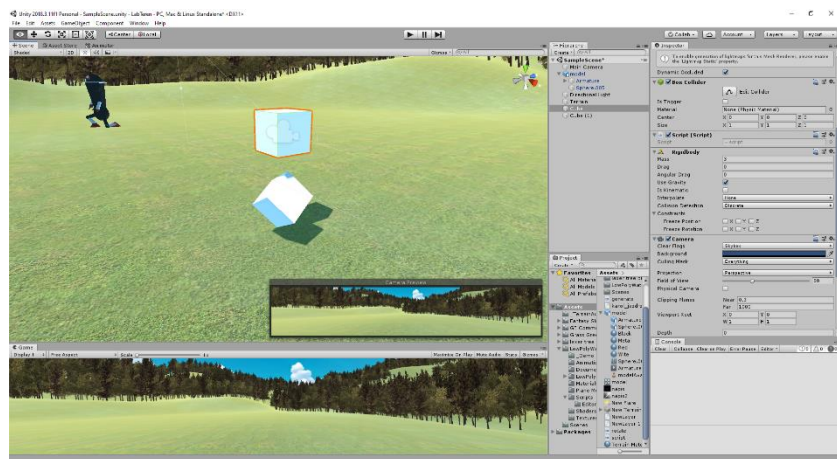
1. Podstawy niezwiązane z fizyką

W pierwszej kolejności zaimportowany został teren i model stworzony na poprzednich zajęciach.



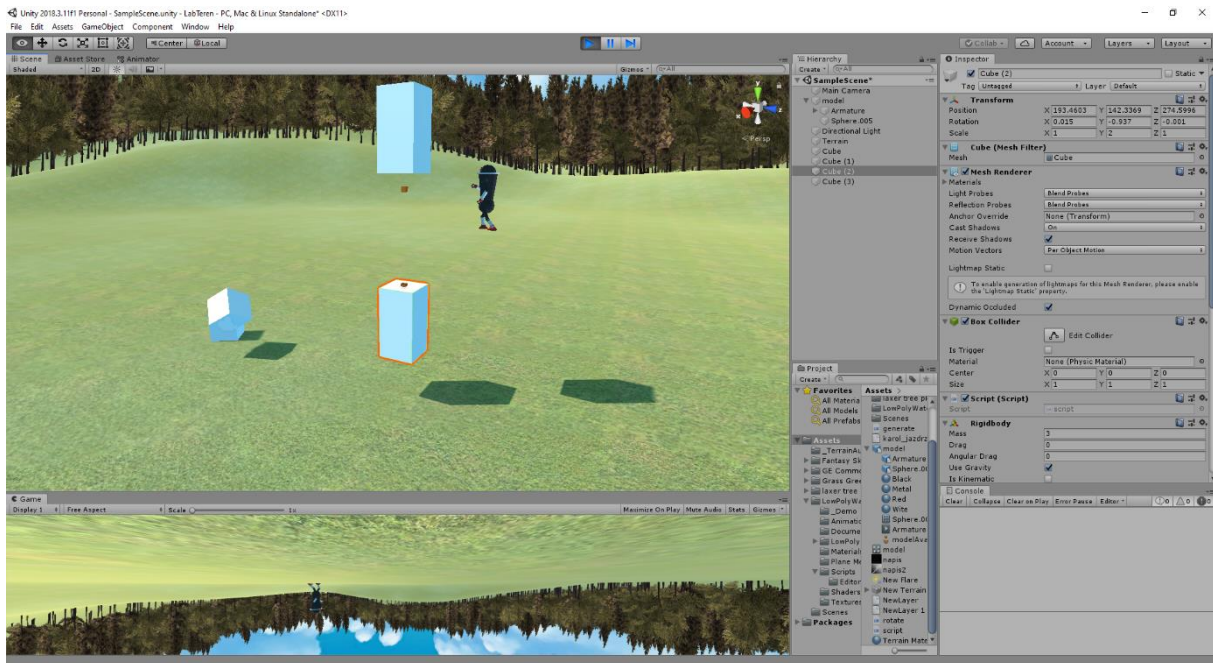
2. Rigidbody

Dodany został obiekt Cube wraz z Rigidbody oraz drugi znajdujący się pod nim nachylony pod kątem. Po uruchomieniu symulacji kostka upada na ziemię.



3. Joints

Dodane zostały dwa prostokątne połączone na początku przy pomocy fixed joint – upadły na ziemię ze stałą odległością od siebie. Następnie, połączenie zostało zamienione na spring joint. Prostokąt na dole opadł ruchem sprężystym.

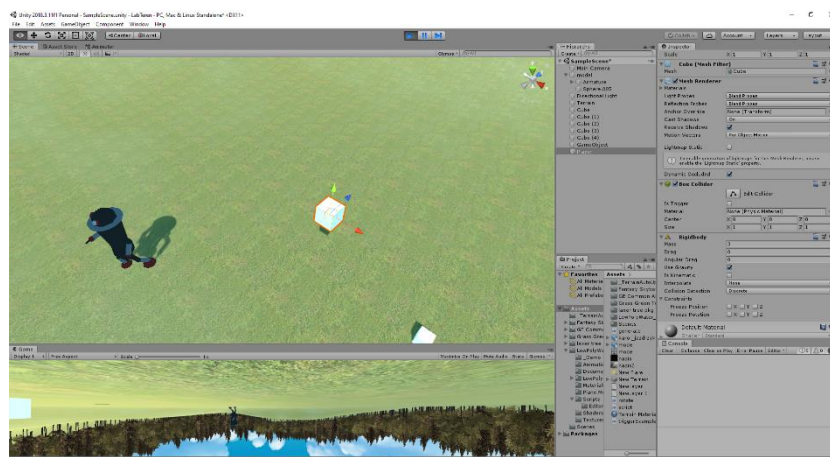


4. Siły

W tej części laboratorium dodano sześcian oraz skrypt nadający mu siły. Na początku symulacji zostaje podrzucona w górę, a następnie obraca się ze stałą prędkością.

```
void Start()
{
    gameObject.GetComponent<Rigidbody>().AddForce(transform.right * 500);
}

void Update()
{
    gameObject.GetComponent<Rigidbody>().AddTorque(transform.up * 200);
    gameObject.GetComponent<Rigidbody>().AddTorque(transform.forward * 20);
}
```



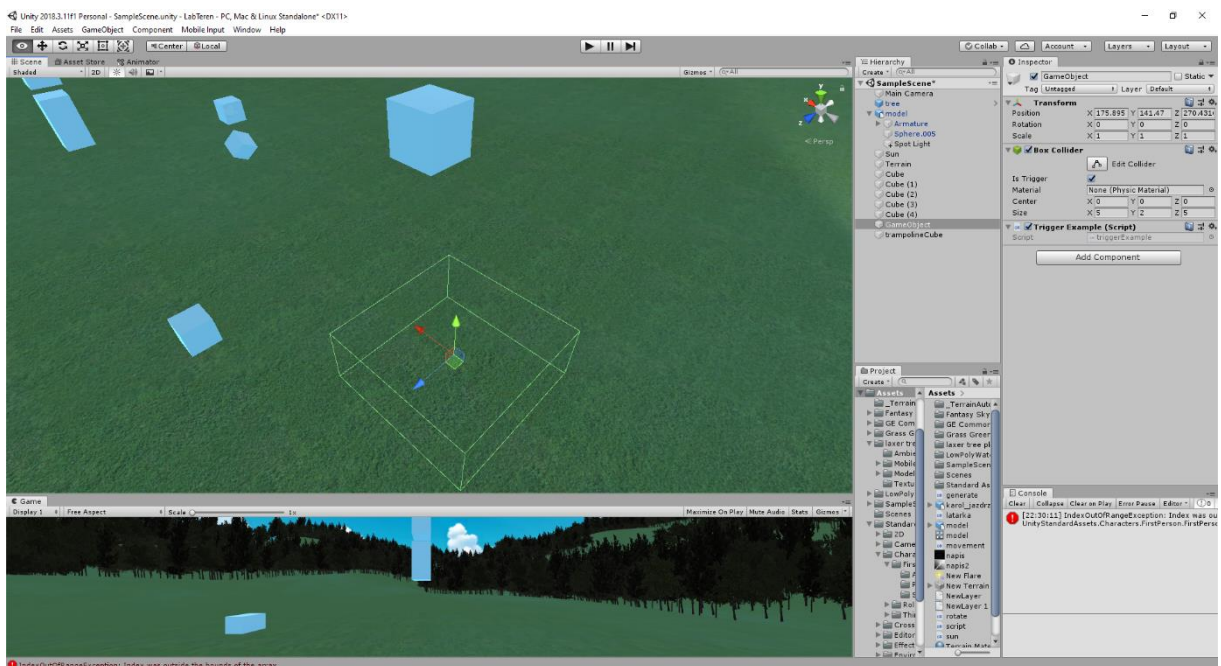
5. Collision i Trigger

Stworzony został kolejny sześcian z tagiem „Player” oraz pusty GameObject zawierający collider i skrypt. Zawierał on następujące funkcje:

```
void OnTriggerEnter(Collider col)
{
    if(col.tag == "Player")
    {
        col.GetComponent<Rigidbody>().AddForce(Vector3.up * 100.0f,
ForceMode.Impulse);
    }
}

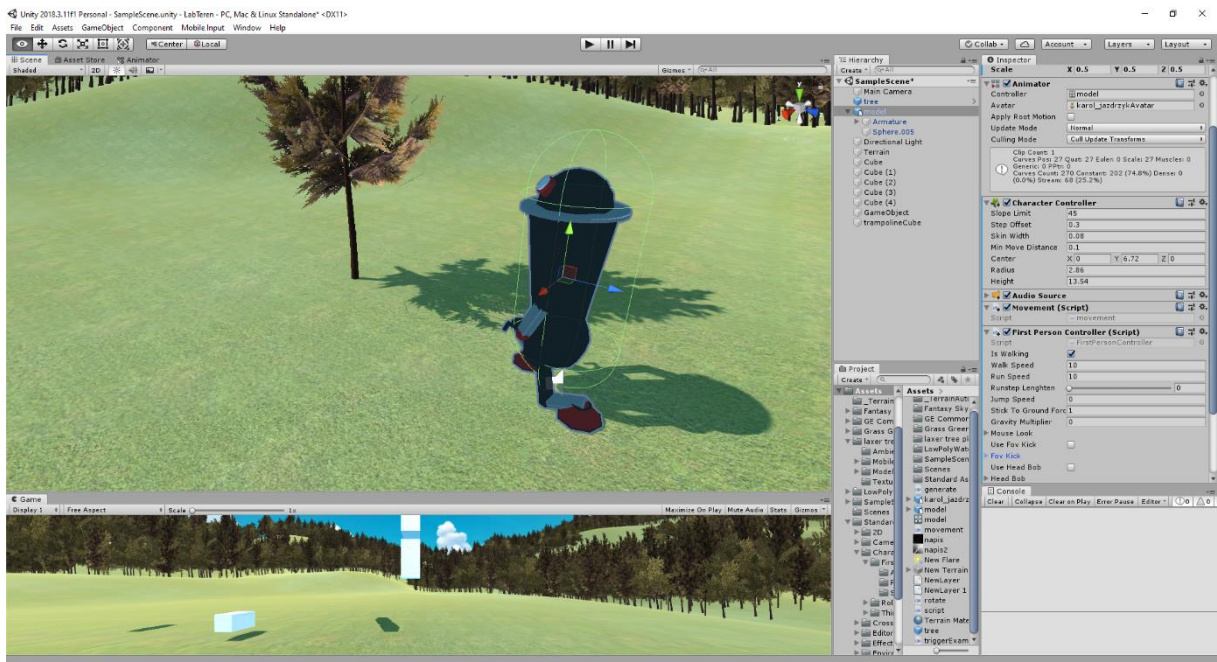
void OnTriggerStay() { }
void OnTriggerExit() { }
```

Po upadku w collider, kostka zostawała ponownie wybita w powietrze obijając się jak na trampolinie.



6. Dodatkowe elementy

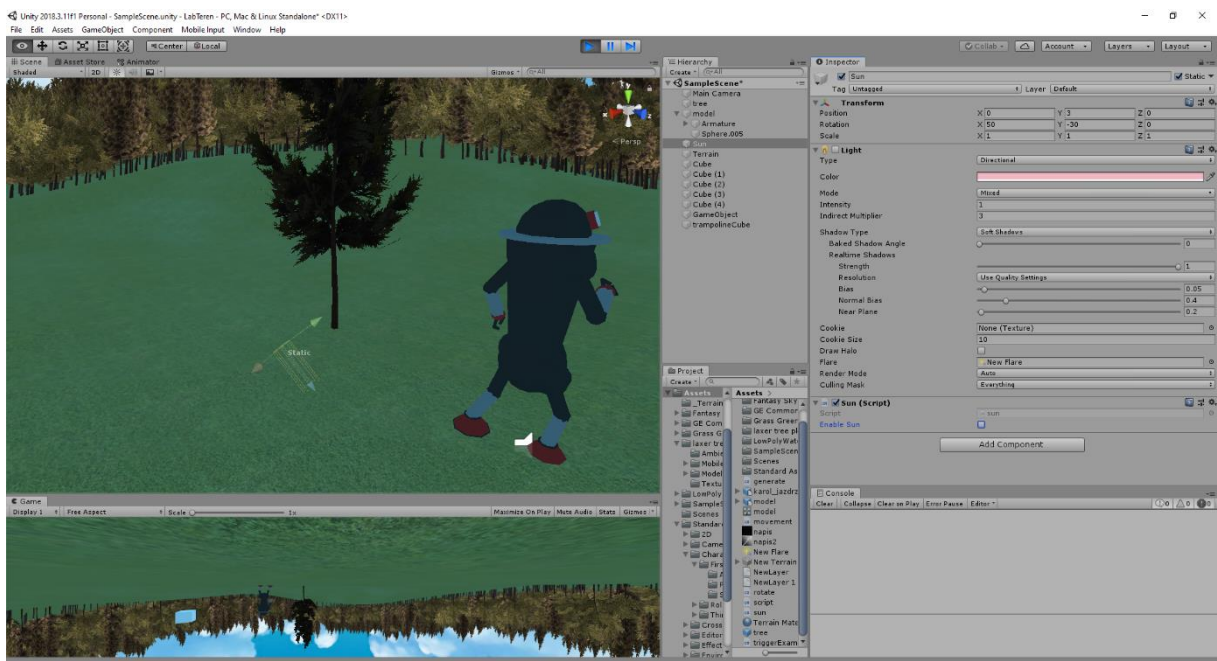
Dodano do postaci FPSCharacterController, collider capsule oraz drzewo wraz z colliderem. Możliwe było poruszanie się postaci z wykorzystaniem gotowego controllera.



Następnie dodano możliwość włączenia i wyłączenia słońca z poziomu GUI:

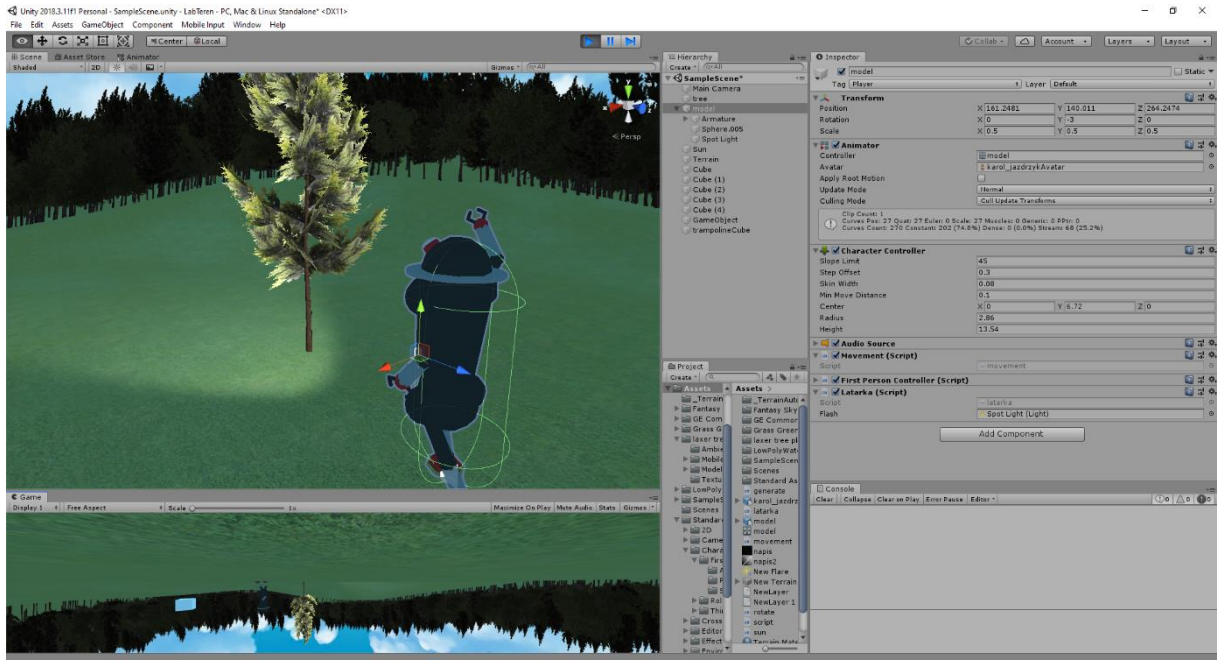
```
public bool enableSun = true;

if(enableSun)
{
    gameObject.GetComponent<Light>().enabled = true;
} else
{
    gameObject.GetComponent<Light>().enabled = false;
}
```



Na końcu zostało dodane światło kierunkowe do postaci jako latarka możliwe do włączenia lub wyłączenia przez klawisz F.

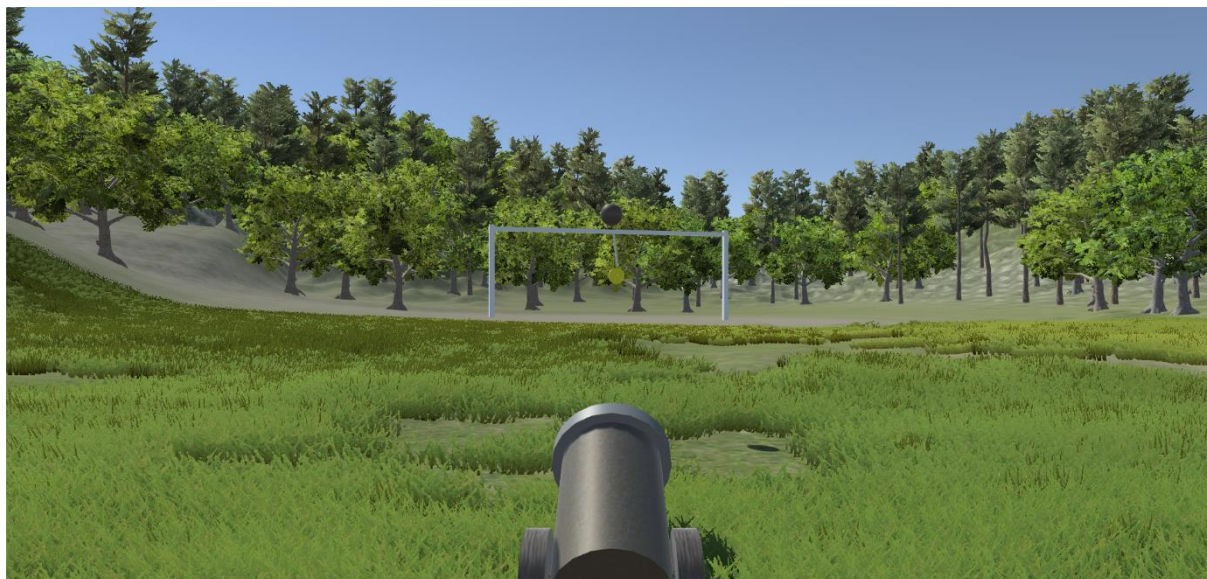
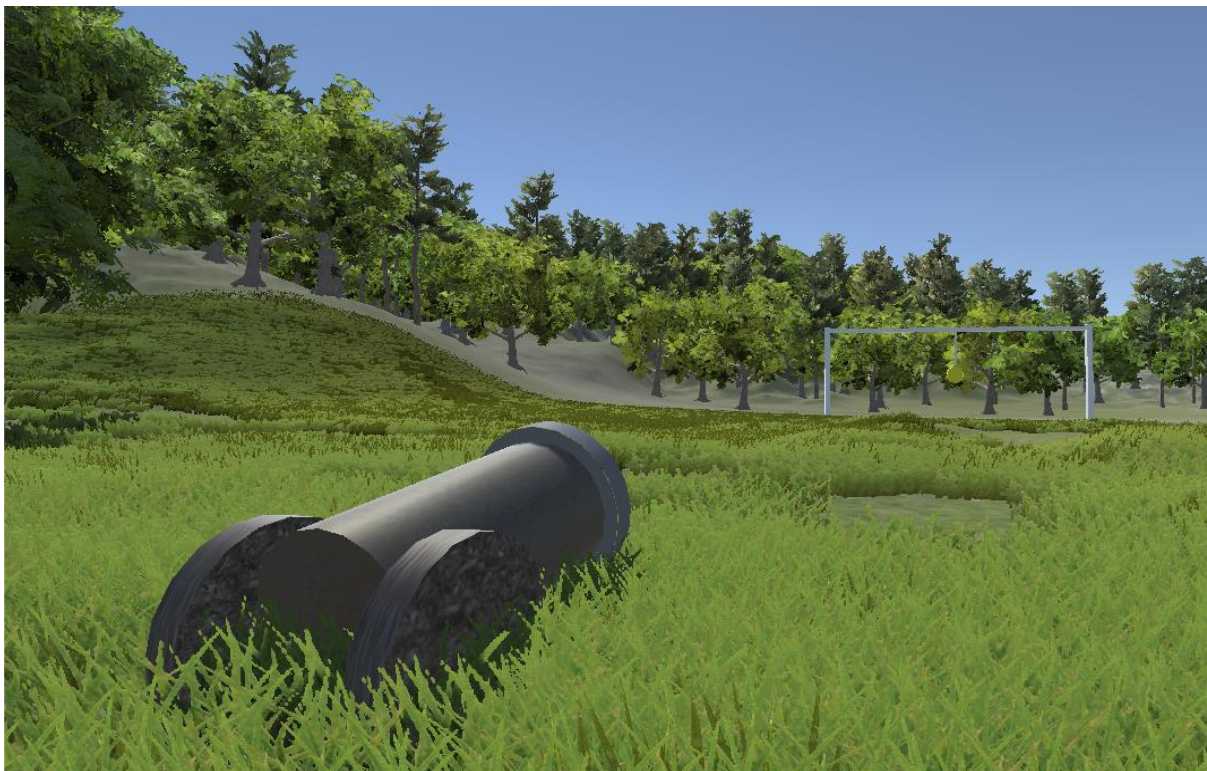
```
void Update()
{
    if (Input.GetKeyDown(KeyCode.F) && flash != null)
    {
        flash.enabled = !flash.enabled;
    }
}
```

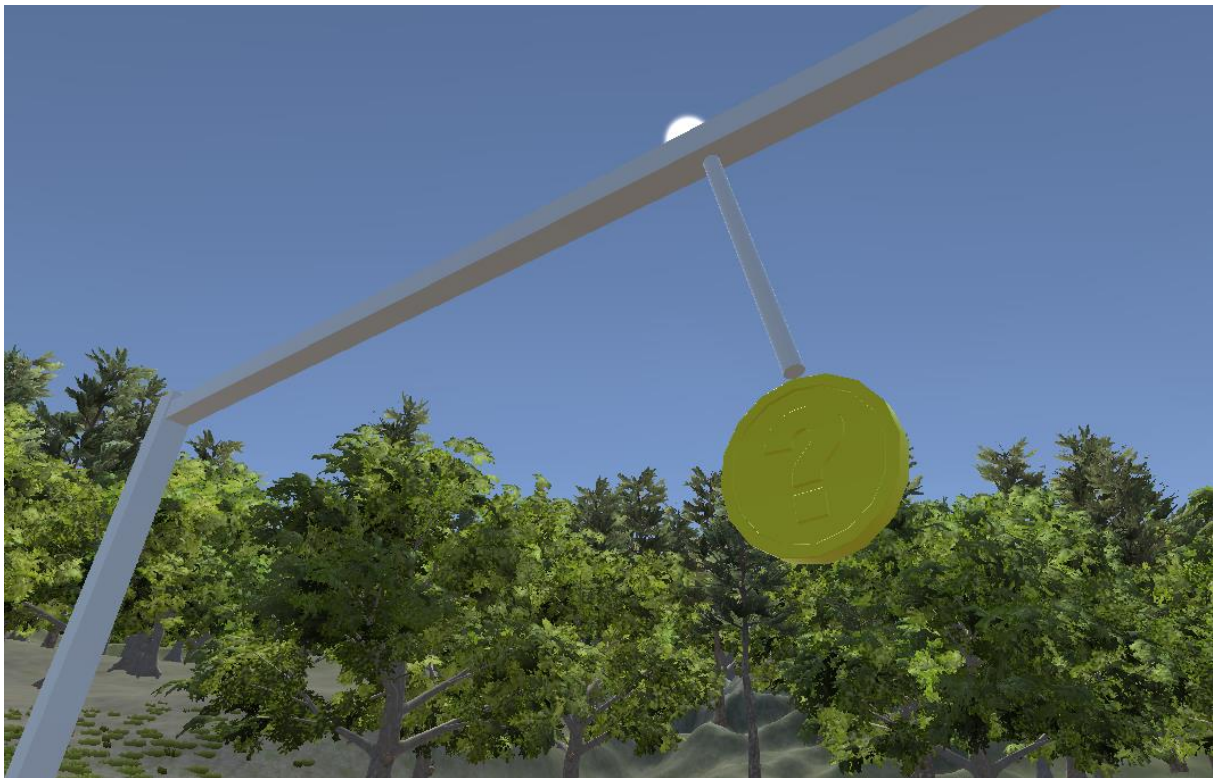


7. Projekt - specyfikacja

7.1. Opis zasad gry

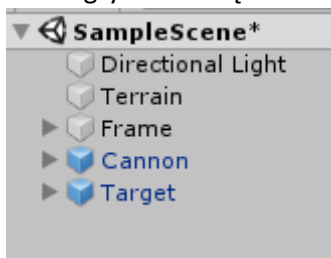
Celem gracza jest trafienie wiszącej tarczy na słupku za pomocą kuli armatniej. Cel samoczynnie porusza się w lewo i w prawo swobodnym ruchem wahadłowym.





7.2. Specyfikacja wewnętrzna

Scena gry składa się z obiektów światła, terenu, ramki na cel, armaty, celu.



Teren, drzewa oraz trawa została stworzona wykorzystując wbudowane narzędzia środowiska Unity. Na rzecz projektu zostały stworzone cztery skrypty. Controller.cs obiektu Cannon zarządza ruchem armaty po wciśnięciu klawiszy oraz tworzy kule:

```
public GameObject ball = null;

void Update()
{
    if (Input.GetKey(KeyCode.A))
    {
        transform.Rotate(.0f, -0.15f, .0f);
    }
    if (Input.GetKey(KeyCode.D))
    {
        transform.Rotate(.0f, 0.15f, .0f);
    }
    if (Input.GetKey(KeyCode.W))
    {
        Transform barrel = transform.Find("barrel");
        if(barrel.rotation.x < -0.2f)
        {
            barrel.Rotate(0.15f, .0f, .0f);
        }
    }
}
```

```

    }
}
if (Input.GetKey(KeyCode.S))
{
    Transform barrel = transform.Find("barrel");
    if (barrel.rotation.x > -0.7f)
    {
        barrel.Rotate(-0.15f, .0f, .0f);
    }
}
if (Input.GetKeyDown(KeyCode.Space))
{
    Transform pos = transform.Find("barrel").Find("ballPosition");
    Instantiate(ball, pos.position, pos.rotation, null);
}
}

```

MoveTarget.cs w obiekcie Target obsługuje poruszanie się celu:

```

void Update()
{
    Transform link = transform.Find("link1");

    link.GetComponent<Rigidbody>().AddForce((float)Math.Sin((float)Time.frameCount /
75.0f)*5.0f, .0f, .0f);
}

```

CheckCollision.cs w obiekcie Target sprawdza czy zaszła kolizja i wyświetla grafikę wygranej w przypadku trafienia:

```

bool winConditionMet = false;
public Texture2D endScreen;

void OnTriggerEnter(Collider col)
{
    if (col.tag == "Player")
    {
        winConditionMet = true;
    }
}

void OnGUI()
{
    if (winConditionMet == true)
    {
        GUI.Label(new Rect(Screen.width/2 - 100,
Screen.height/2 - 50, 200, 100), endScreen);
    }
}

```

Skrypt Launch.cs obiektu Ball nadaje siłę kuli przy utworzeniu przez armatę i usuwa ją 8 sekund po wystrzeleniu:

```

float lifespan = 0;
void Start()
{
    lifespan = Time.time;
    GetComponent<Rigidbody>().AddForce(transform.TransformDirection(new
Vector3(0, 10000, 0)));
}

```

```

    }

    void Update()
    {
        if(Time.time - lifespan > 8.0f)
        {
            Destroy(this.gameObject);
        }
    }
}

```

Wykorzystane zostały assety z kolekcji standardowych assetów Unity. Obiekty na scenie zbudowane są z prostych brył pokrytych teksturą.

7.3. Specyfikacja zewnętrzna

Obrót armaty obsługiwany jest za pomocą klawiszy A oraz D. Klawisze W i S podnoszą oraz obniżają lufę. Strzał wykonywany jest przy pomocy Spacji. Po trafieniu w cel pokazywana jest tekstura na ekranie informująca o wygranej. Obrót armaty możliwy jest w dowolną stronę, a lufę można odchylić w osi pionowej w zakresie 50 stopni.



7.4. Testy

Wykonano testy wszystkich funkcjonalności. Kontrola armaty była zgodna z oczekiwaniami, umiejscowienie kamery nie przeszkadzało w rozgrywce, kule były usuwane po 8 sekundach od wystrzelenia oraz po trafieniu była wykrywana kolizja w celu wyświetlenia ekranu końcowego.

7.5. Podsumowanie

Projekt pozwolił na przyswojenie wiadomości z kształtowania terenu oraz fizyki w środowisku Unity. Wykorzystane zostały 3 części fizyki (kolizje i trigger do trafienia celu, joints w celu zawieszenia tarczy oraz nadanie siły w celu wystrzelenia kuli).

Link do github:

<https://github.com/karo1404/ZTPGK-Projekt>