SPRAWOZDANIE

Zajęcia: Grafika i Multimedia

Prowadzący: prof. dr hab. inż. Vasyl Martsenyuk

|  |  |
| --- | --- |
| Laboratorium Nr 2  Data 14.10.2023  Temat: "Ćwiczenia w modelowaniu  materiałów o rożnych właściwościach  Fizycznych"  Wariant 3 | Adam Kubliński  Informatyka  II stopień, niestacjonarne, zaoczne  I semestr, gr.1A |

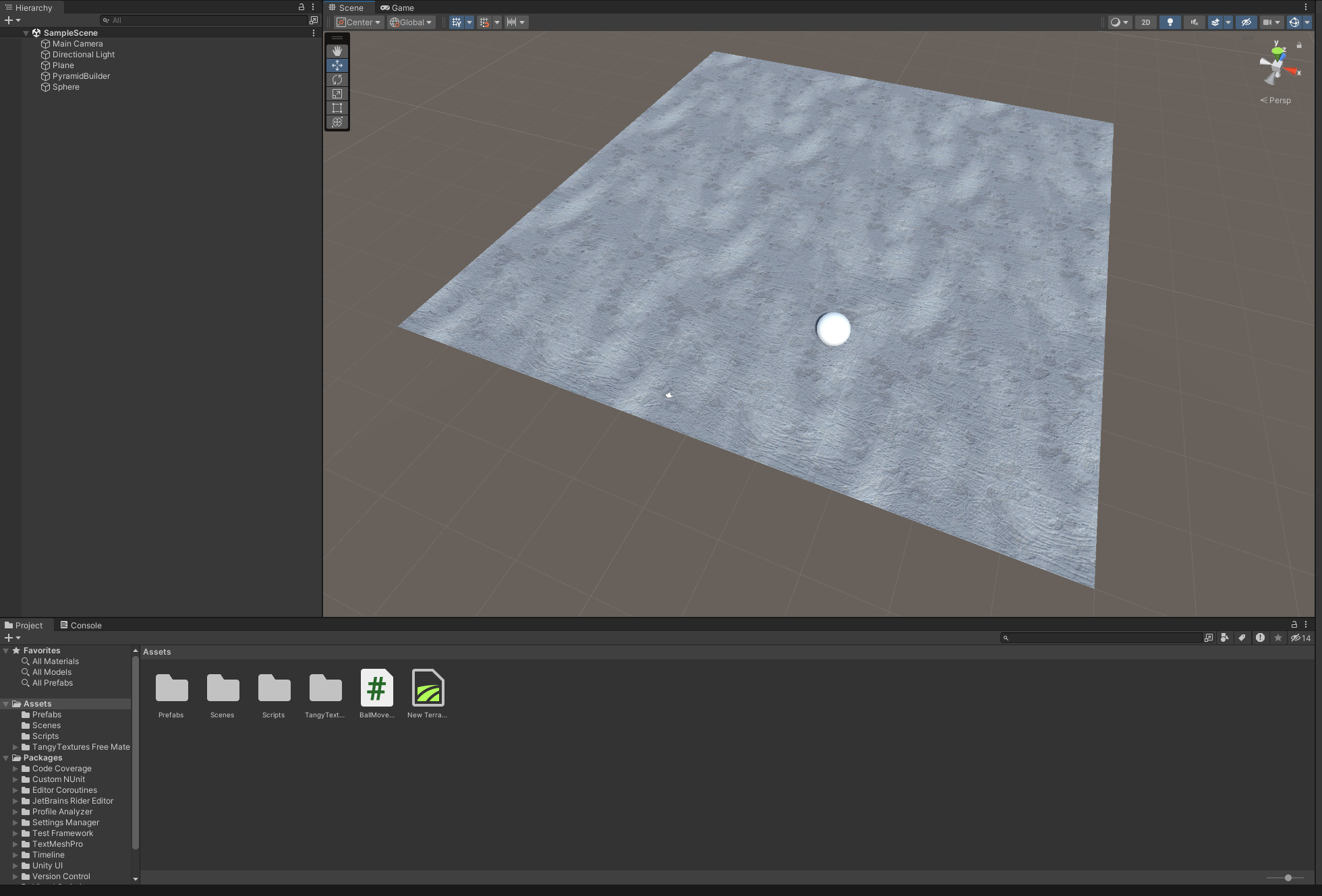
# 1. Polecenie: wariant 3 zadania

* Program 1. Narysować piramidę z użyciem elementów podstawowych wg wariantu: 3. Cylinder
* Ilość poziomów piramidy jest 10.
* Piramida jest położona na powierzchni ziemi. Użyć materiały dla powierzchni ziemi (paczka Tangy): 3. concrete
* Połączyć elementy podstawowe piramidy sprężynami (Spring Joint)
* Opracować grę w której gracz za pomocą kuli chce rozbijać piramidę (używając Constant Force)

# 2. Opis programu opracowanego (kody źródłowe, rzuty ekranu)

## 2.1. Obszar gry i gracz

Stworzono środowisko oraz obszar gry z materiałem concrete. Stworzono gracza w postaci kuli i napisano skrypt, który umożliwia kontrolę nad nim za pomocą strzałek.



Rysunek 1: Obszar gry wraz z graczem

## 2.2. Skrypt nadany graczowi:

using UnityEngine;

public class BallMovement : MonoBehaviour

{

public float moveSpeed = 7.0f; // Prędkość ruchu kuli

private Rigidbody rb;

void Start()

{

rb = GetComponent<Rigidbody>();

}

void Update()

{

float moveHorizontal = Input.GetAxis("Horizontal");

float moveVertical = Input.GetAxis("Vertical");

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

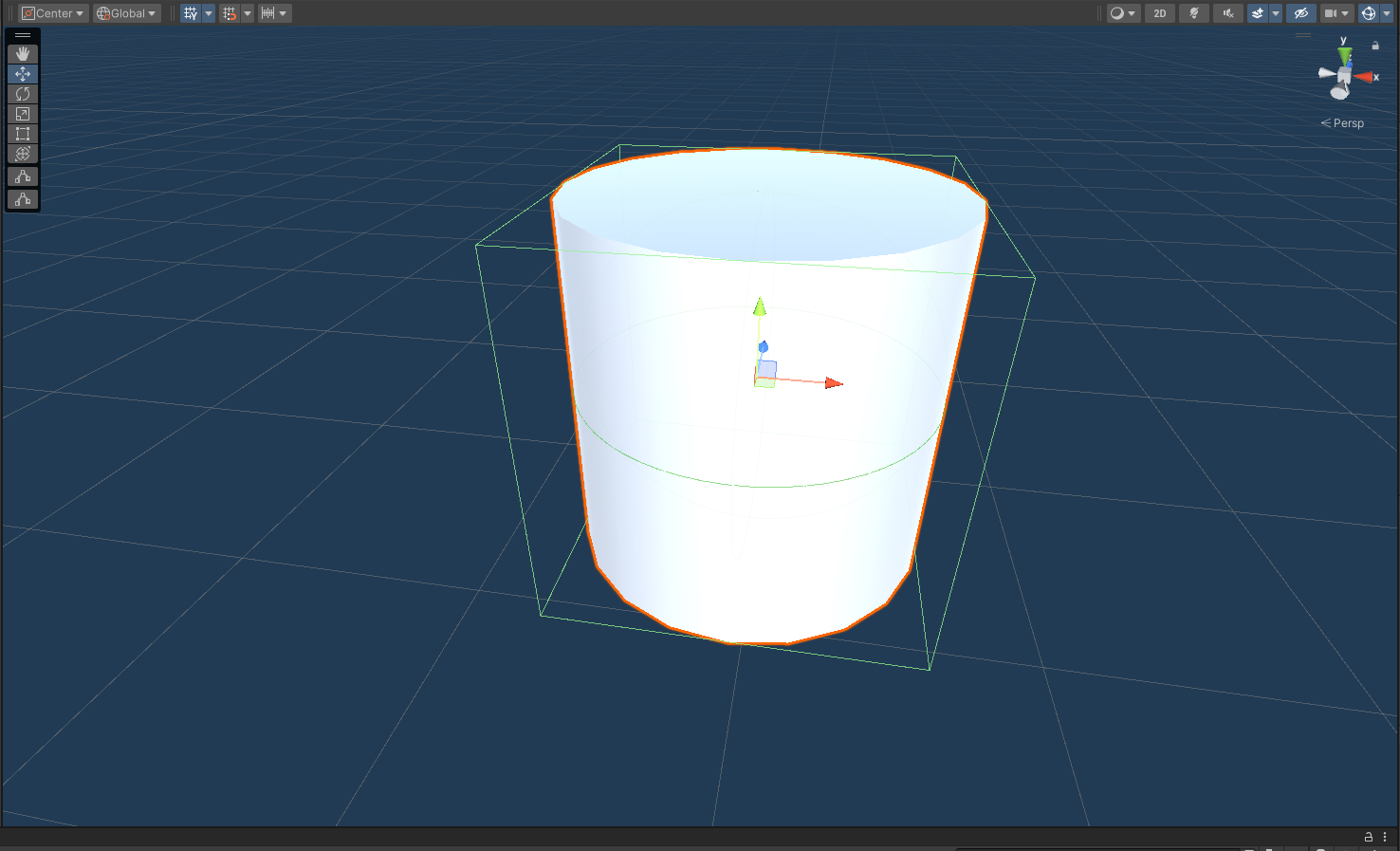
rb.velocity = movement \* moveSpeed;

}

}

## 2.3. Piramida

Piramida zostaje stworzona za pomocą stworzonego Prefabu “Cylinder”



Rysunek 2 Prefab „Cylinder”

Następnie zostaje napisany skrypt „PyramidBuilder” w języku C#, który tworzy piramidę z Cylindrów na wysokość 10 pięter:

using UnityEngine;

public class PyramidBuilder : MonoBehaviour

{

public GameObject prefab; // Przypisz prefab, który chcesz użyć, w Inspektorze

public int initialSize = 5; // Początkowy rozmiar kwadratu (ilość prefabów na każdym boku)

public float spacing = 1.0f; // Odległość między prefabami

public float offset = 1.0f; // Przesunięcie na osi X i Z

void Start()

{

Vector3 startPosition = transform.position;

int size = initialSize;

GameObject previousPrefab = null;

for (int floor = 0; floor < initialSize; floor++)

{

for (int i = 0; i < size; i++)

{

for (int j = 0; j < size; j++)

{

float xOffset = i \* spacing + floor \* offset;

float zOffset = j \* spacing + floor \* offset;

Vector3 prefabPosition = new Vector3(startPosition.x + xOffset, startPosition.y + (floor \* spacing), startPosition.z + zOffset);

GameObject currentPrefab = Instantiate(prefab, prefabPosition, Quaternion.identity);

if (previousPrefab != null)

{

ConnectWithSpringJoint(previousPrefab, currentPrefab);

}

previousPrefab = currentPrefab;

}

}

size--;

}

}

void ConnectWithSpringJoint(GameObject prefab1, GameObject prefab2)

{

SpringJoint springJoint = prefab1.AddComponent<SpringJoint>();

springJoint.connectedBody = prefab2.GetComponent<Rigidbody>(); // Sprawdź, czy prefab ma komponent Rigidbody

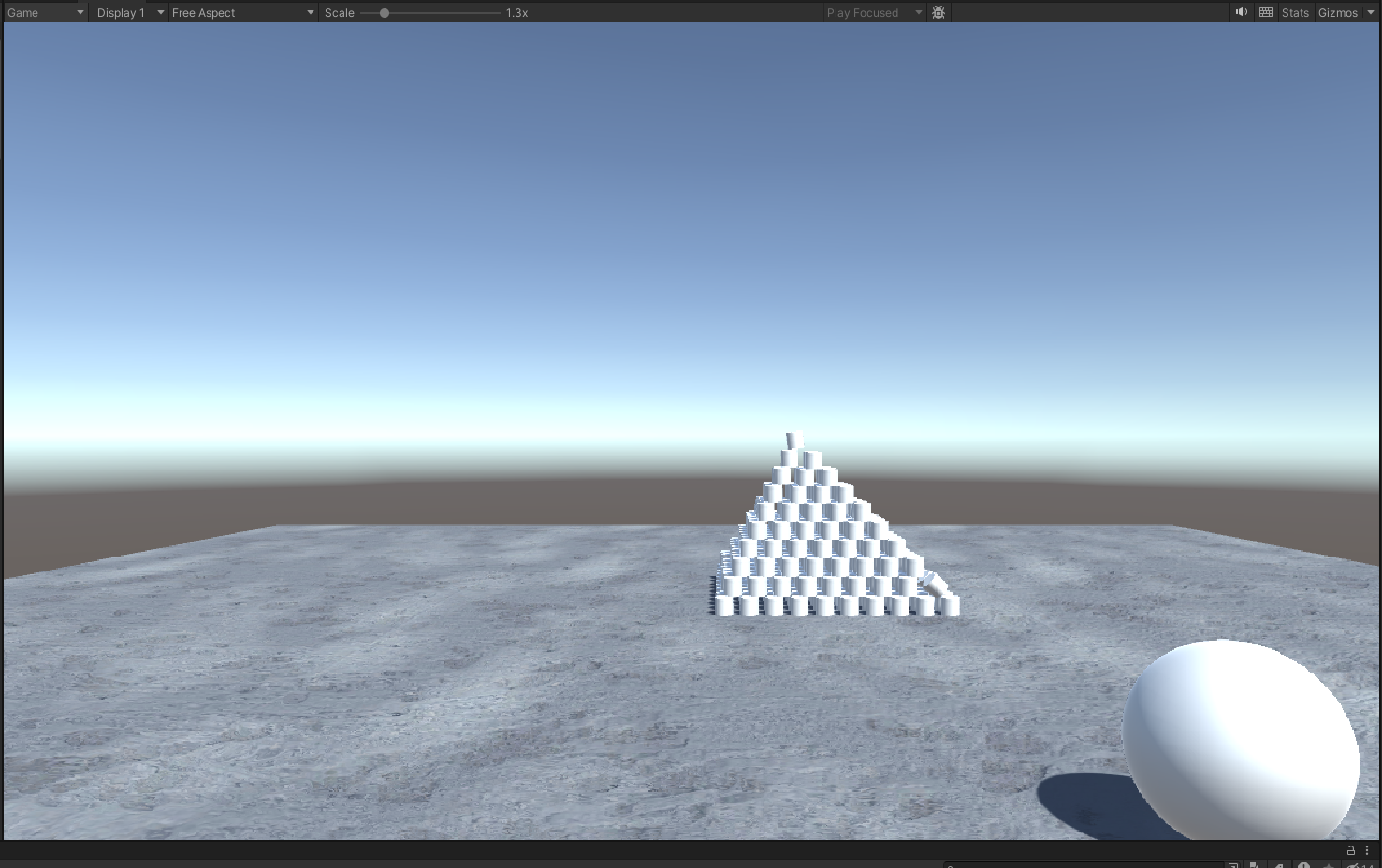
springJoint.spring = 500.0f; // Dostosuj wartość sprężystości, jeśli to konieczne

springJoint.damper = 5.0f; // Dostosuj tłumienie, jeśli to konieczne

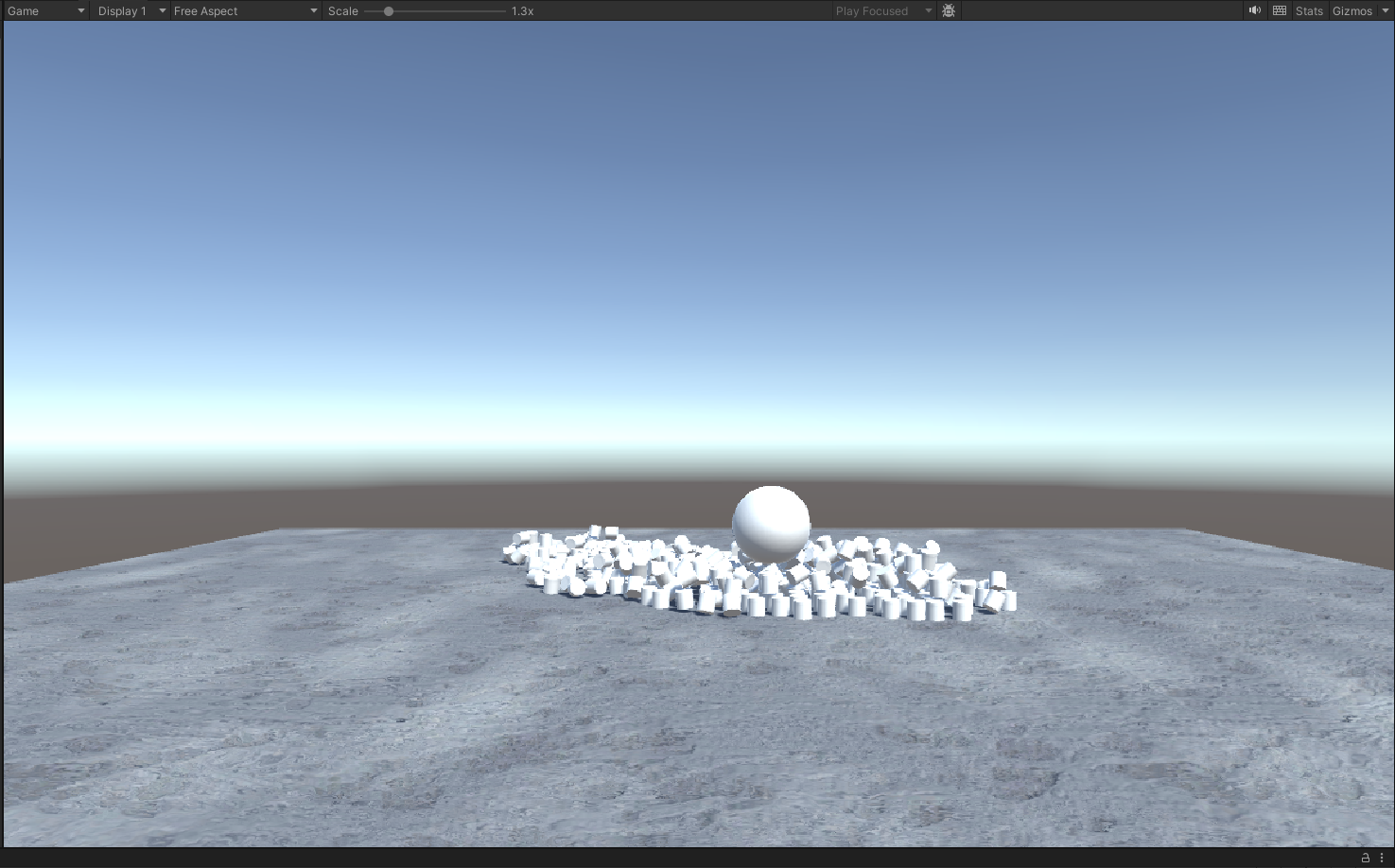
}

}

Efektem uruchomienia skryptu jest stworzenie piramidy o zadanych parametrach w chwili uruchomienia gry.



Rysunek 3: Gra w chwili rozpoczęcia z wygenerowaną piramidą



Rysunek 4: Efekt kolizji piramidy z graczem (kula)

Pliki z grą zostały udostępnione w repozytorium GitHub: https://github.com/Adamadacho/GIM-Collider-Game.git

3. Wnioski

Gra została stworzona w oparciu o instrukcję jak i doświadczenie z poprzedniego laboratorium. Na podstawie zdobytej wiedzy stworzono grę, w której jest możliwe zaobserwowanie działania na siebie różnych obiektów zdefiniowanych w środowisku gry. Zapoznano się również z metodą generowania struktur złożonych z jednakowych prefabów za pomocą skryptu.