Вступ

Комплекс лабораторних робіт модуля «Розробка комп'ютерних ігор за допомогою Unity 3D» виконується відповідно до програми навчальної дисципліни «Технології розробки крос-платформних комп'ютерних ігор» та передбачає поглиблення теоретичних знань з навчальної дисципліни та набуття практичних навичок роботи з Unity 3D за допомогою спеціалізованого ігрового обладнання.

До комплексу лабораторних робіт входять індивідуальні завдання, які передбачають проектування та розробку власного ігрового додатку та ігрового 3D-контенту в Unity 3D. Тема індивідуального завдання вибирається студентом самостійно. Індивідуальні завдання виконуються студентом самостійно з консультацією викладача.

Максимальна сумарна оцінка за виконання лабораторних робіт в межах навчальної дисципліни складає 6 балів. У кожній роботі 30% нараховується за виконання практичної частини і її відповідність поставленій задачі і 70% – за захист роботи (аргументоване пояснення ходу роботи і відповіді на теоретичні питання).

Виконання та презентація фінального індивідуального завдання оцінюється у 29 балів.

Графік проведення поточного оцінювання

Номер тижня	Оцінювання
2	Оцінка виконання лабораторної роботи 1
4	Оцінка виконання лабораторної роботи 2
6	Оцінка виконання лабораторної роботи 3
8	Оцінка виконання лабораторної роботи 4
10	Оцінка виконання лабораторної роботи 5
12	Оцінка виконання лабораторної роботи 6
16	Оцінка виконання індивідуального завдання

Подання звіту щодо виконання лабораторних робіт

Лабораторні роботи виконуються протягом аудиторного заняття. У кінці кожного заняття студент зобов'язаний продемонструвати виконане завдання.

У межах двох тижнів студент зобов'язаний самостійно оформити звіт до попередньої лабораторної роботи, де повинні бути вказані основні кроки, що були виконані у ході даної роботи.

Протягом аудиторного заняття, відведеного для виконання наступної лабораторної роботи студент має захистити попередню роботу, відповівши на

питання щодо теоретичного матеріалу та надавши пояснення про хід і результати лабораторної роботи.

Оцінювання лабораторних робіт:

− за кожну лабораторну роботу студент отримує 6 балів (максимальна сумарна оцінка за усі лабораторні роботи в межах навчальної дисципліни складає 36 балів);

− якщо робота виконана не самостійно, то знімається 50% від максимальної кількості балів;

− якщо в програмі не витримано основні правила створення програмних продуктів (наприклад, модульність, дружній інтерфейс, наявність коментарів) знімається 5%.

За кожен тиждень запізнення захисту лабораторної роботи (комп'ютерного практикуму) нараховується штрафні – 0,2 бали (максимально 5 днів).

Методи оцінки змістовних модулів навчальної дисципліни

Кількість балів в загальній оцінці змістовних модулів відповідає наступному:

Виконання лабораторної роботи 1	максимально 6 балів.
Виконання лабораторної роботи 2	максимально 6 балів.
Виконання лабораторної роботи 3	максимально 6 балів.
Виконання лабораторної роботи 4	максимально 6 балів.
Виконання лабораторної роботи 5	максимально 6 балів.
Виконання лабораторної роботи 6	максимально 6 балів.
Виконання контрольної роботи	максимально 35 балів.
Виконання індивідуального завдання	максимально 29 балів.

Усі набрані бали підсумовуються (максимально 100 балів), штрафні бали за запізнення в представленні звіту з лабораторної роботи віднімаються.

Обладнання

Комплекс лабораторних робіт модуля «Розробка комп'ютерних ігор за допомогою Unity 3D» виконується в ігровій навчальній лабораторії «GameLab».

За призначенням ігрову навчальну лабораторію «GameLab» поділено на чотири робочі зони:

1. Зона розробки та тестування ігор в середовищі Android, яка містить робочі місця з наступною комп'ютерною технікою:

- 15 робочих місць студентів для розробки ігор, кожне з яких оснащене ноутбуком HP Pavilion 15-ab246ur, ігровою клавіатурою Logitech G103 та маніпулятором Logitech F310;
- 5 спеціалізованих робочих місць, кожне з яких оснащене планшетом Lenovo Tab 3 Business X70F, обладнанням ігрової віртуальної реальності Razer Virtual Reality (OSVR) ecosystem&headset, шоломами віртуальної реальності EMOTIV Insight 5 Channel Mobile EEG + USB Receiver та ігровою гарнітурою TrackIr 5 Premium Head Tracking for Gaming.

2. Зона розробки та тестування ігор в середовищі Apple iOS, яка містить 2 спеціалізованих робочих місця, кожне з яких оснащене комп'ютером Apple A1418 iMac 21.5" та планшетом Apple A1566 iPad Air 2.

3. Зона запису, обробки та прослуховування аудіоконтенту гри, яка оснащена звуковим USB-інтерфейсом 2x4 BEHRINGER U-Phoria UMC204HD з мікрофонами BEHRINGER, кабелями PROEL BULK250LU5, мікрофонними стійками PROEL RSM180 та навушниками Sennheiser HD558 для якісного запису аудіоігрового контенту.

4. Зона розробки фото- та відеоігрового контенту гри, яка оснащена зеркальною фотокамерою Canon EOS700D 18-55mm, сканером Epson Perfection V370 Photo, лазерним принтером Canon i-SENSYS LBP6030W для друку через WiFi та з мобільних пристроїв.

Сервер HP ProLiant ML350 Gen9 забезпечує безперебійне функціонування мережі Wi-Fi навчальної лабораторії «GameLab».

Крім того, ігрова лабораторія «GameLab» оснащена ігровими приставками для тестування ігор Sony PlayStation 4 500GB та XBOX 360 Slim 250Gb Kinect.

В ігровій навчальній лабораторії «GameLab» переважно використовується програмне забезпечення з відкритим кодом, що передається у вільне користування для студентів та викладачів для некомерційної експлуатації.

Це насамперед крос-платформні середовища та засоби розробки додатків для операційних систем Android, iOS, Windows з використанням мов програмування Java, JavaScript, Python, C++.

Але для створення якісного ігрового контенту залучено також комерційне програмне забезпечення, для якого відсутні якісні вільно розповсюджувані аналоги, а саме:

- крос-платформні професійні графічні програми для обробки та покращення растрових і векторних зображень, фотографій, графічного дизайну (Adobe Photoshop CC ALL Multiple Platforms);
- професійні графічні програми моделювання високополігональних цифрових скульптур і текстурного фарбування 3D моделей для створення цифрових 3D об'єктів і 2D скетчів з широким спектром текстур і фарб (Autodesk Mudbox);
- спеціалізоване програмне забезпечення для якісної 3D-анімації, рендеринга, моделювання та візуалізації із застосуванням відеоефектів (Autodesk Maya 2016);
- спеціалізоване програмне забезпечення для підготовки, компонування, редагування, запису та завантаження аудіоконтенту високої якості (Sound Forge Audio Studio 10);
- спеціалізоване програмне забезпечення для підготовки, компонування, редагування, запису та завантаження відеоконтенту різної роздільної здатності та якості (Corel VideoStudio Pro X9 ML).

Таким чином, обладнання та програмне забезпечення ігрової навчальної лабораторії «GameLab» відповідає сучасним вимогам підготовки фахівців з розробки крос-платформних комп'ютерних ігор та ігрового контенту.

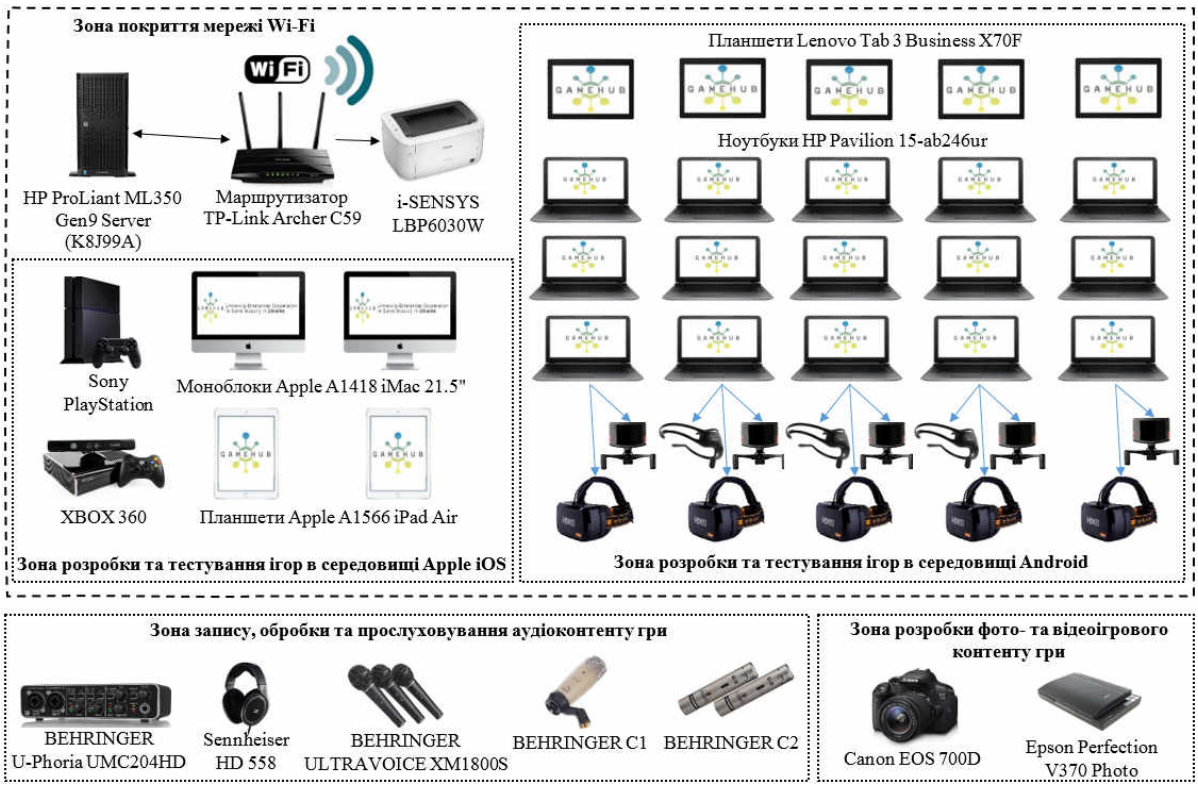

Рисунок 1.1 - Концепція ігрової навчальної лабораторії «GameLab»

ЛАБОРАТОРНА РОБОТА № 1
ТЕМА: «РЕЖИМИ РОБОТИ ТА ІНТЕРФЕЙС UNITY 3D. РОБОТА З ГРАФІКОЮ»

Анотація

Лабораторна робота орієнтована на ознайомлення студентів з інтерфейсом системи Unity 3D (головне меню, огляд проекту, ієрархія, сцена, ігровий вид, інспектор) та методами роботи з графікою (освітлення, камери, матеріали, текстури, ландшафти, рендеринг).

Мета лабораторної роботи

Освоїти основні прийоми роботи з 2D- і 3D-режимами системи Unity 3D, інтерфейсом системи та методами роботи з графікою. Вивчити можливості роботи з освітленням в Unity 3D (Directional Light, Point Light, Spot lights, Area Light). Вивчити можливості роботи з камерою в Unity 3D (Perspective and orthographic cameras). Освоїти основні прийоми роботи з матеріалами та текстурами в Unity 3D. Вивчити можливості створення та редагування ландшафтів (Terrain). Освоїти основні прийоми рендерингу в Unity 3D.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде знати та вміти застосовувати основні прийоми роботи з 2D- і 3D-режимами системи Unity 3D, вміти застосовувати основні графічні компоненти системи для роботи з освітленням та камерою, застосовувати основні прийоми роботи з матеріалами та текстурами при розробці ігрових сцен, створювати та редагувати ландшафти, вміти застосовувати прийоми рендерингу при розробці ігрових додатків.

Режими роботи та інтерфейс Unity 3D

Робоче вікно Unity 3D розбите на 6 взаємопов'язаних областей (рис. 1.1). (За замовчуванням в Unity 3D включено вид «**Default**» або «**Tall**». Якщо у вас інший вид робочого вікна, то можете переключити його в меню «**Layout**» у верхньому правому кутку вікна).

Перелік всіх робочих областей вікна Unity 3D подано у табл. 1.1.

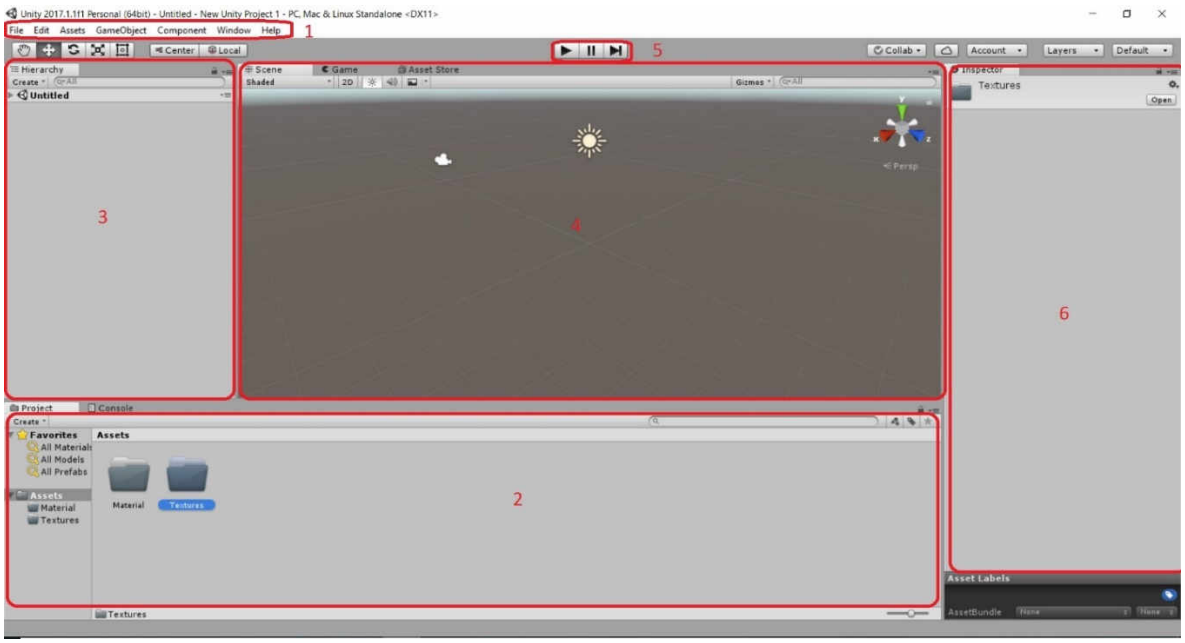

Рисунок 1.1- Робоче вікно Unity 3D

Таблиця 1.1 - Перелік робочих областей вікна Unity

Перелік робочих областей	**Призначення робочих областей**
1. Main menu (Головне меню)	Рядок тексту зверху, де розташовуються всі команди, доступні в програмі. Багато команд продубльовано кнопками і меню в робочих областях, тому головне меню не обов'язково використовувати.
2. Project View (Огляд проекту)	Список, в якому показані всі файли, які використовуються в грі: файл сцени, файл програмного коду, графічні і аудіофайли.
3. Hierarchy (Ієрархія)	Список об'єктів, які додано на сцену. Тут можна працювати з об'єктами, копіювати їх, перейменовувати, видаляти.
4. Scene (Сцена)	Область, де відображається ігровий світ або ігрова сцена. Тут ми можемо додавати нові об'єкти, перетягувати їх, змінювати вид.
5. Game (Гра)	Область попереднього перегляду, де видно, як сцена буде виглядати в грі. Тут можна налаштовувати різні налаштування екрану і режиму відео.
6. Inspector (Інспектор)	Список, що складається з декількох різних по виду розділів. Показує всі властивості обраного об'єкта: розміри, моделі, текстури, скрипти.

Головне меню Unity 3D

Головне меню Unity 3D стандартно розташовується у верхньому лівому кутку вікна (рис. 1.2).

У головному меню містяться взагалі всі команди програми, а в основних областях вікна вони лише дублюються.

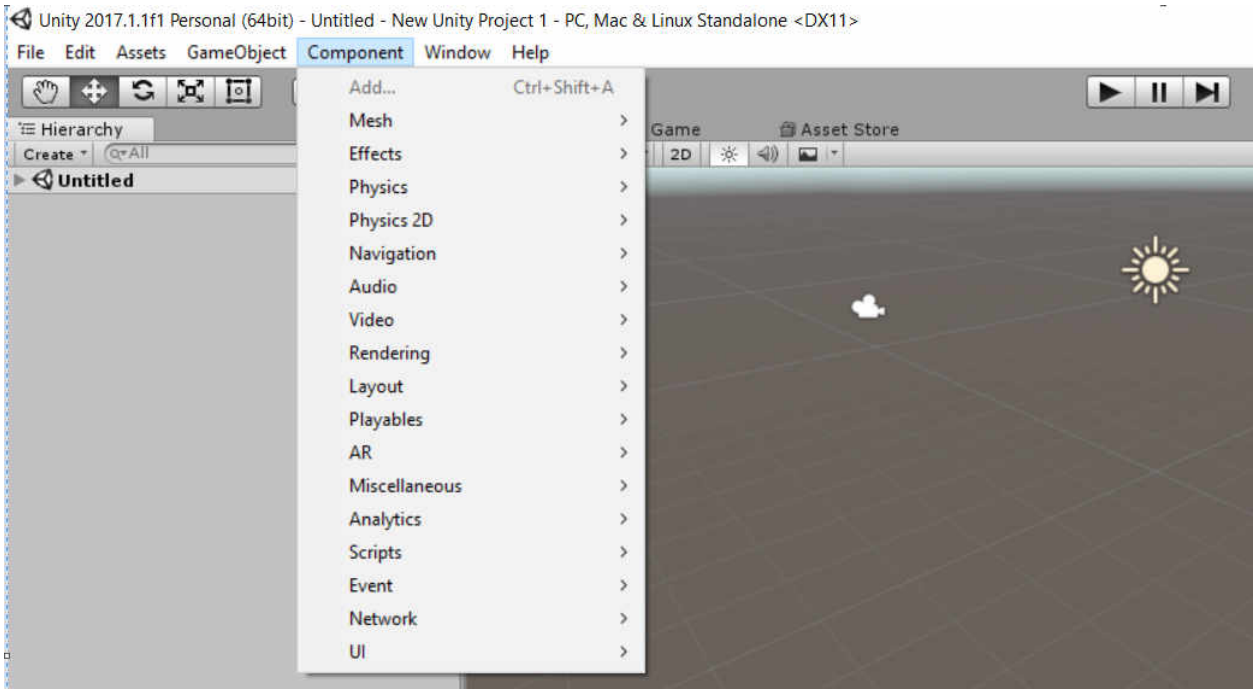

Рисунок 1.2- Головне меню Unity 3D

Огляд проекту (Project View)

Кожен проект містить папку «Assets». Вміст цієї папки представлено в області «Project View» (рис. 1.3). Це ресурси гри: файли-скрипти, 3D-моделі, текстури, аудіофайли, префаби (об'єкти, які можна клонувати).

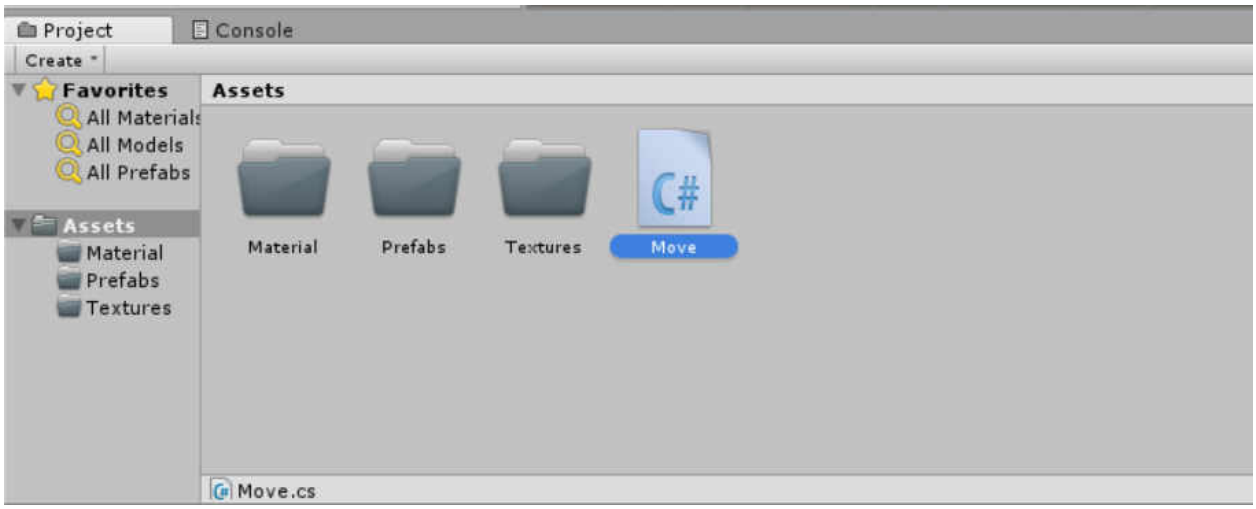

Рисунок 1.3- Область «Project View». Файли проекту

Переміщення файлів можна виконувати через стандартний **Провідник Windows**, вибравши «Reveal in Exlorer». Але при переміщенні ресурсів в **Провіднику Windows** будуть втрачені всі посилання між об'єктами.

Краще завжди переміщати файли всередині проекту тільки в **«Project View»**.

Додати новий ресурс в проект можна двома способами:

Перетягнути файл з **Провідника Windows** в область **«Project View»**.

Виконати команду **«Assets»** → **«Import New Assets»**.

Проекти гри складаються з одного або декількох файлів сцени. Кожна окрема сцена - це окремий рівень гри. Сцени так само зберігаються в папці «Assets» і відображаються в «Project View».

Деякі ігрові ресурси не є файлами, і створюються безпосередньо в Unity. Для створення ресурсу використовується меню **«Create»** (рис. 1.4).

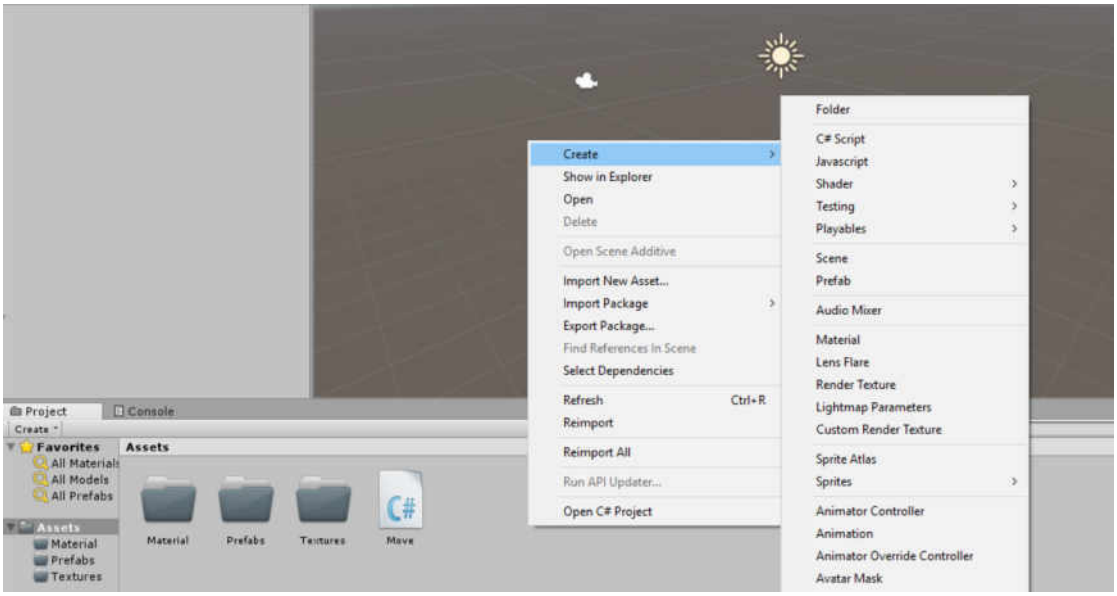

Рисунок 1.4 - Меню «Create»

Меню **«Create»** дозволяє додавати в проект скрипти, префаби, папки та інше. Будь-який ресурс або папку можна перейменувати, натиснувши **«F2»** або зробивши два кліка по імені. Якщо затиснути кнопку **«Alt»**, то при розкритті директорії будуть розкриті і всі піддиректорії.

Ієрархія (Hierarchy)

Ієрархія містить всі об'єкти (GameObject) відкритої сцени. Об'єкти, що додаються в сцену або видаляються з неї, відображаються або навпаки перестають відображатися в Hierarchy (рис. 1.5).

Спадкування (Parenting). В ієрархії Unity об'єктів можна задавати успадкування. Будь-який об'єкт може бути дочірнім по відношенню до іншого. Дочірній об'єкт буде рухатися і обертатися разом з батьківським об'єктом. Для створення дочірнього зв'язку досить перетягнути необхідний об'єкт на батьківський об'єкт в Hierarchy.

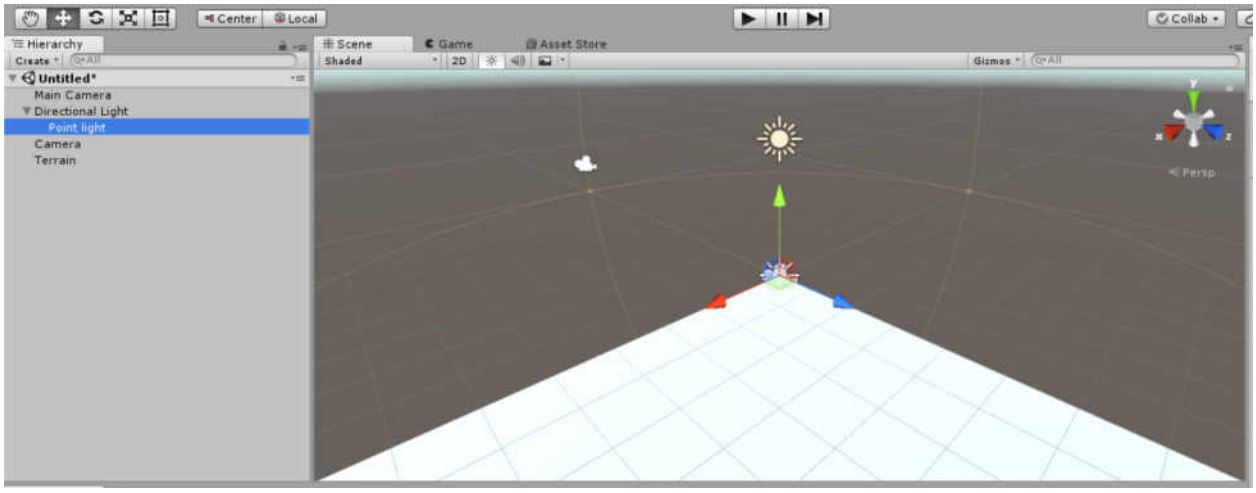

Рисунок 1.5 – Ієрархія об'єктів

Сцена (Scene View)

Ігрова сцена використовується для розстановки об'єктів (оточення, персонажі, камери, системи частинок та інше). Сцена може бути ігровим рівнем, головним меню, заставкою (рис. 1.6).

В 2D режимі управління є інтуїтивно зрозумілим. У 3D режимі є безліч прийомів для переміщення по сцені:

1. Затиснута права кнопка миші (ПКМ) активує режим вільного польоту.
2. Переміщатися можна клавішами WASD на манер гри в жанрі FPS.
3. Виберіть об'єкт у списку ієрархії і натисніть «F». Вид сцени буде центровано і масштабовано по вибраному об'єкту.

4. Затиснуті клавіша Alt та ліва кнопка миші (ЛКМ) буде крутити камеру навколо поточної точки опори.

5. Затиснуті клавіша Alt та середня кнопка миші (СКМ) будуть переміщати камеру.

6. Затиснуті клавіша Alt та ПКМ будуть масштабувати вигляд сцени.

7. Застосувати альтернативний режим переміщень можна за допомогою клавіші Q.

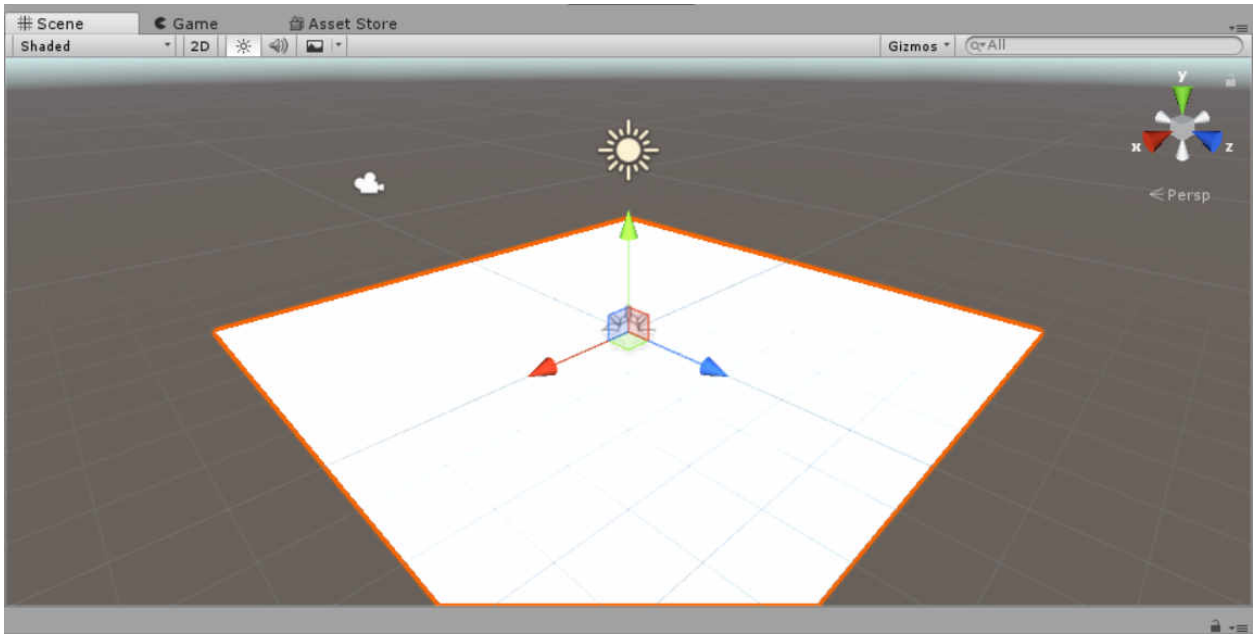

Рисунок 1.6 - Ігрова сцена

Ігровий вид (Game View)

Game View - попередній огляд гри (рендер з ігрової камери). У розташуванні вікон за замовчуванням **«Game View»** відсутній. Для його включення потрібно вибрати вкладку **«Game»** над ігровою сценою (рис. 1.7).

Play Mode

Три кнопки у верхній частині вікна Unity 3D відповідають за управління попереднім оглядом гри: **«Play», «Pause»** і **«Step».** Всі зміни, зроблені під час попереднього перегляду, скидаються при виході з нього. Винятком є зміни в префабах.

Game View Control Bar

Перше меню, що випадає в **«Game View»** - це контроль пропорцій зображення (**Aspect Drop-down**). На деяких дисплеях це співвідношення відрізняється від стандартного 4:3 (наприклад, на широкоформатних моніторах - 16:10).

Далі йде кнопка «**Maximize on Play**». Якщо вона натиснута, то **«Game View»** розтягується на всі вікна редактора при попередньому перегляді.

Кнопка **«Gizmos»** включає відображення контейнерів **Gizmo** в **«Game View».**

Остання кнопка - **«Stats»**. Вона показує статистку рендеринга (**Rendering Statistics**), корисну при оптимізації.

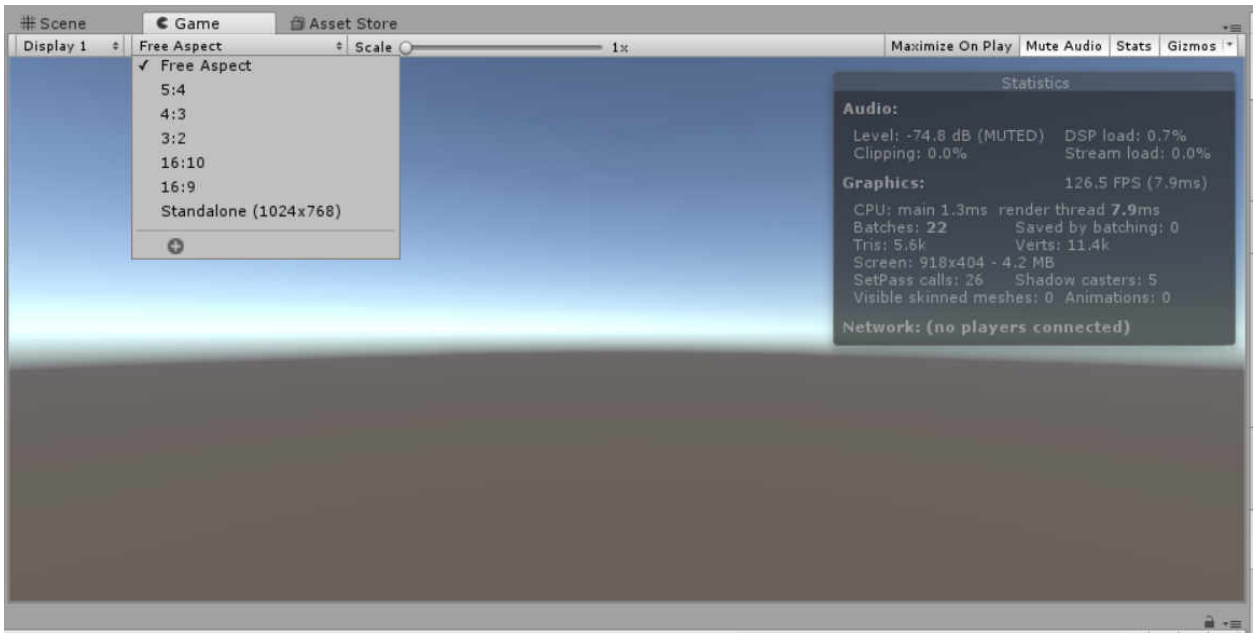

Рисунок 1.7 - Game View

Інспектор (Inspector)

GameObject - це будь-який об'єкт в грі. Сам по собі об'єкт не виконує дій. Йому потрібні спеціальні властивості, перш ніж він стане персонажем, оточенням або візуальним ефектом.

Об'єкти - це лише контейнери. Вони можуть містити в собі різні елементи, комбінації яких перетворюють об'єкт в персонаж, декорацію, спецефект. Ці елементи називаються компонентами (Components). Залежно від того, що потрібно створити, об'єкту присвоюються різні комбінації компонентів.

Перелік об'єктів міститься в області «**Hierarchy**». Якщо в списку ієрархії ми виберемо будь-який об'єкт, то в області «**Inspector**» відобразяться всі властивості цього конкретного об'єкта (рис. 1.8). Тут же можна редагувати всі ці властивості або додавати нові.

Об'єкт може містити в собі такі типи компонентів: розташування в просторі (**Transform**), меши (**meshes**), скрипти (**scripts**), звуки, джерела світла (**Lights**) та інші елементи.

Status Bar

Рядок стану (**Status Bar**) розташований у нижній частині вікна редактора. Він відображає помилки компіляції. Якщо виникають проблеми з

грою, варто заглянути в рядок стану. Подвійне натискання на ньому відкриє вікно консолі (**Console**), в якому відображаються всі помилки (рис. 1.9).

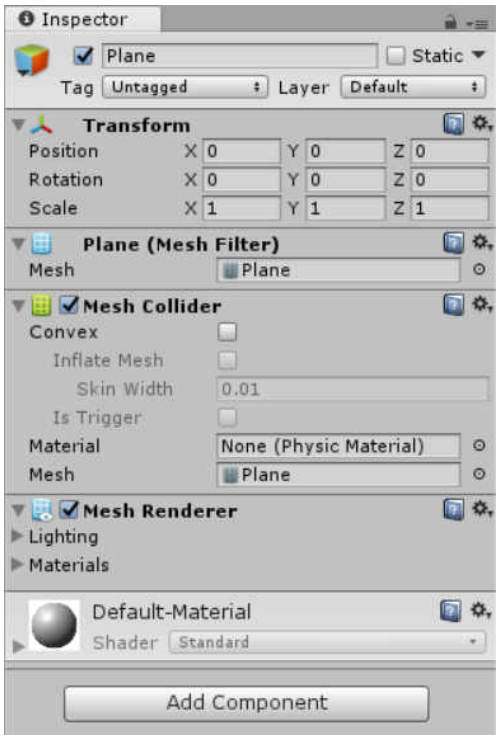

Рисунок 1.8 - Властивості об'єкта в області «Inspector»

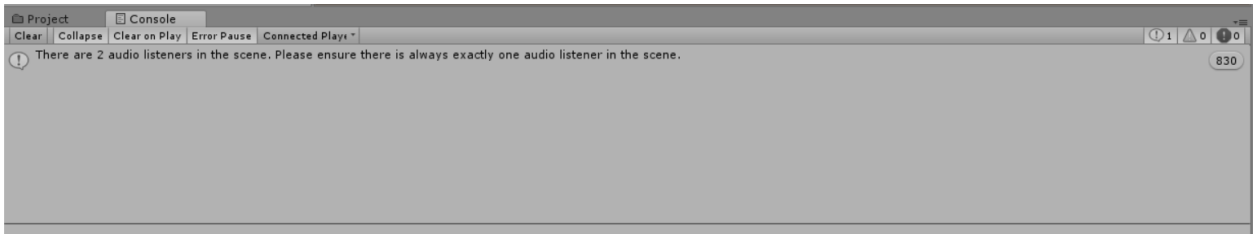

Рисунок 1.9 - Вікно консолі (**Console**)

Налаштування робочого вікна

Можна налаштовувати розташування (Layout) секторів, перетягуючи їх за закладки. Якщо перетягнути закладку в область закладок вже існуючого вікна, то вона буде додана до присутніх там закладок. Також можна прикріпити сектор до краю екрану або краю іншого сектора.

Сектор може бути додано до однієї зі сторін існуючого вікна.

Закладки можуть відкріплює від головного вікна редактора і включатися до складу плаваючого вікна редактора. Плаваюче вікно може містити сектора і закладки так само, як і головне вікно.

Плаваюче вікно редактора схоже на головне вікно, але не має панелі інструментів (Toolbar).

Коли розташування секторів задано, його можна зберегти і завантажити в потрібний момент через меню Layout (Save і Load).

Створення проекту Unity. Імпорт необхідних ресурсів.

Запускаємо Unity.
Обираємо на вкладці **On Disk** пункт **New Unity Project** (рис. 1.10).

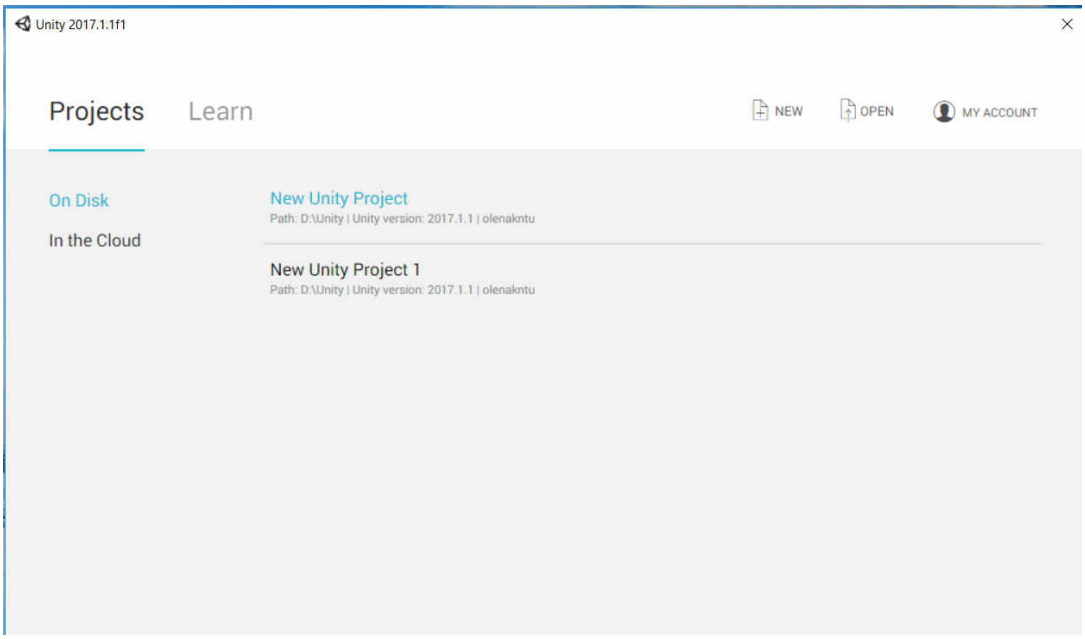

Рисунок 1.10 - Створення нового проекту Unity

Для імпорту необхідних ресурсів (матеріали, текстури) вибираємо меню «Assets → Import package → Custom package» і переходимо в папку з імпортованими ресурсами (рис. 1.11).

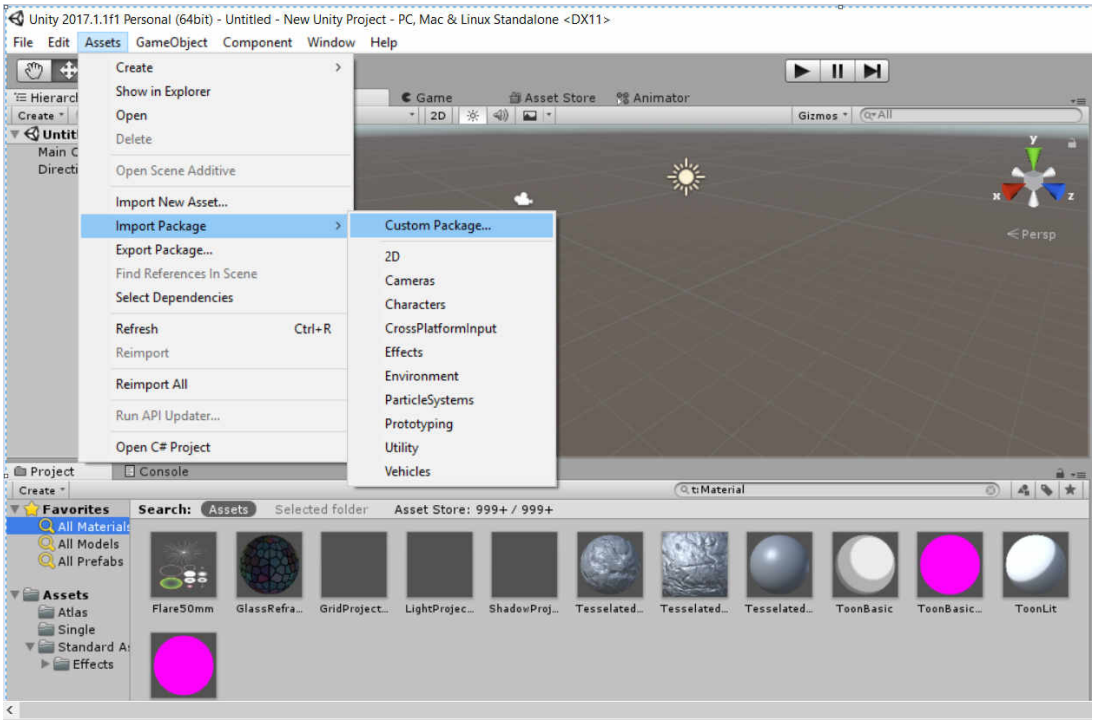

Рисунок 1.11 - Імпорт необхідних ресурсів

6. Генерація ландшафту

Для створення ландшафту в Unity використовується 3D Object «Terrain».

Для створення ландшафту необхідно додати в вікно «Scene» 3D Object **«Terrain»** вибравши меню **«Game Object →3D Object → Terrain»** (рис. 1.12).

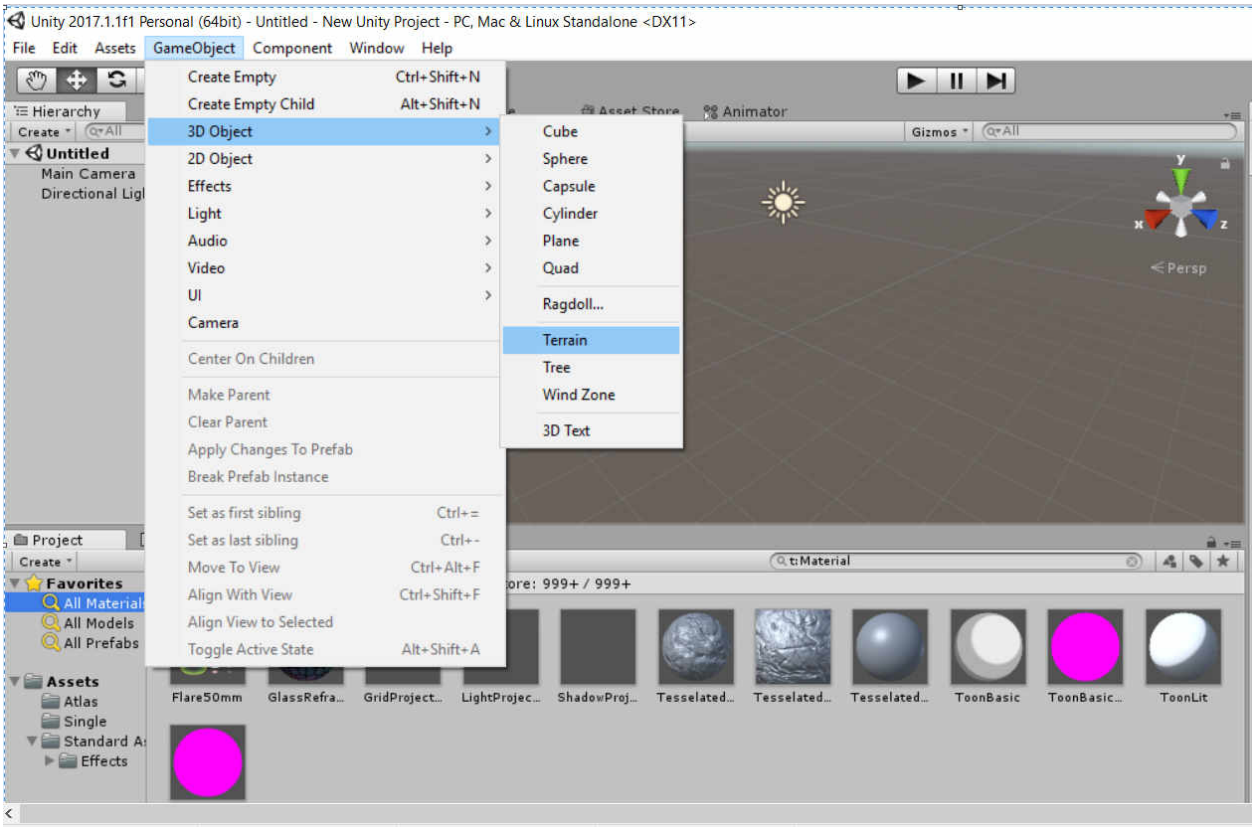

Рисунок 1.12 - Створення 3D Object **«Terrain»**

При цьому, ландшафт спочатку буде просто плоскою рівниною. Однак, якщо ви подивитеся на інспектор при виділеному об'єкті **Terrain**, ви побачите, що там представлений ряд інструментів, які ви можете використовувати для створення будь-яких потрібних вам ландшафтів (рис. 1.13).

Отже, ландшафт готовий. Задамо йому текстуру. В інспекторі об'єктів натискаємо на пензлик а потім на кнопку **Edit Textures**. Обираємо пункт **Add Texture**.

Відкриється вікно додавання текстур до об'єкту **Terrain** «**Add Terrain Texture»** (рис. 1.14).

У вікні «**Add Terrain Texture»** натискаємо «**Select Texture 2D**» .

У вікні «**Select Texture 2D**» вибираємо текстуру, нажимаємо на кнопку **«Add»** та додаємо її до об'єкту **Terrain** (рис. 1.15).

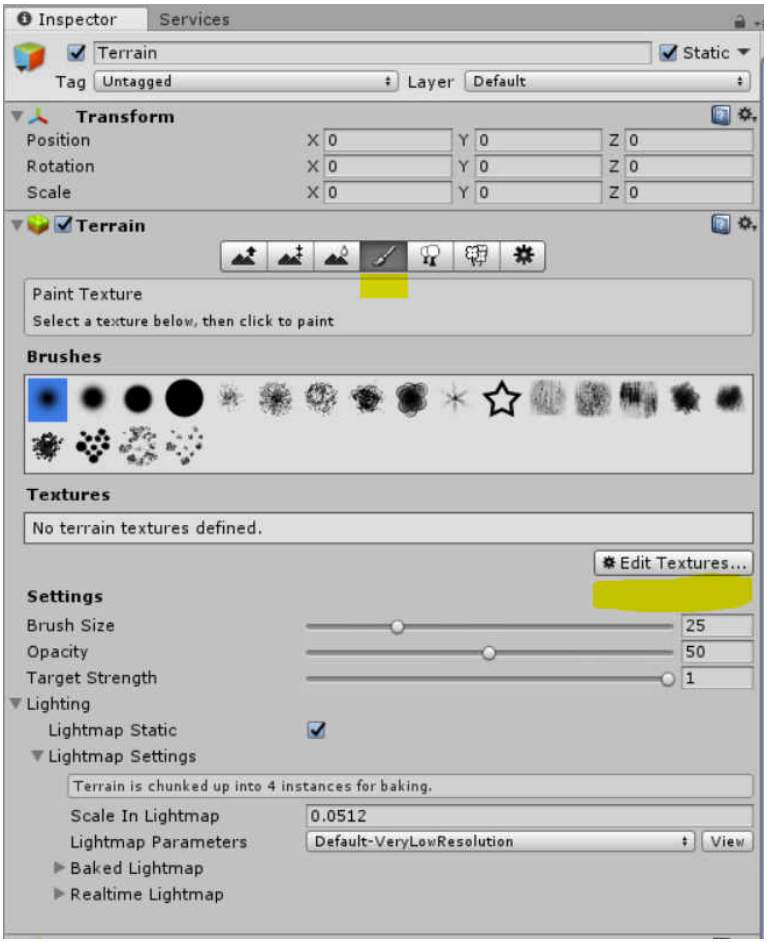

Рисунок 1.13 - Інструменти для створення ландшафтів

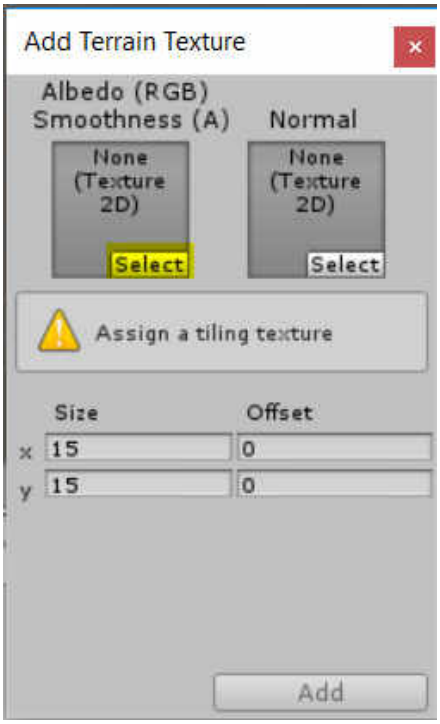

Рисунок 1.14 - Вікно додавання текстур до об'єкту **Terrain** «**Add Terrain Texture**»

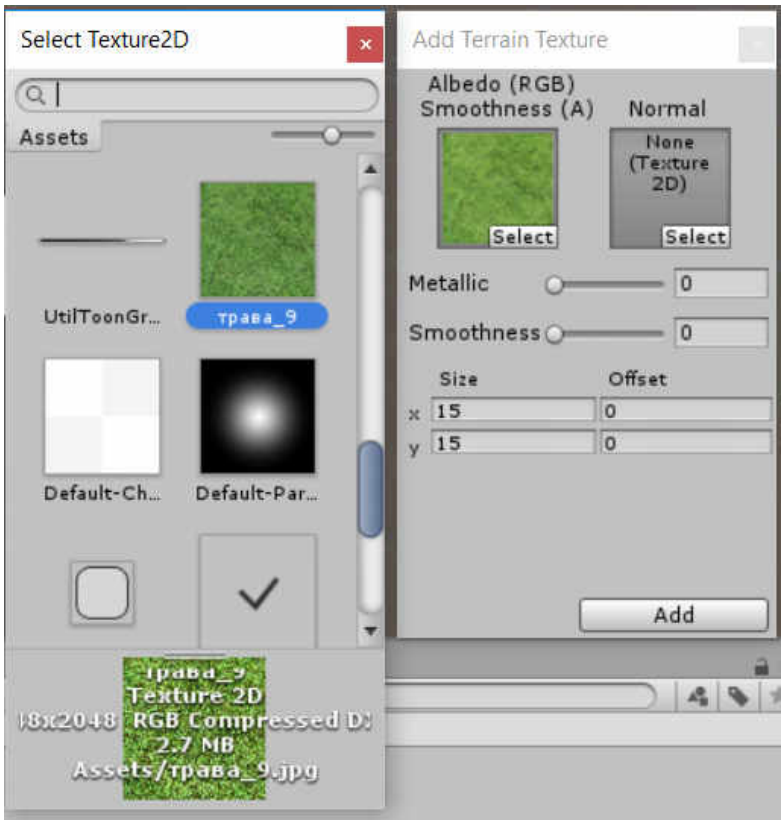

Рисунок 1.15 – Вікно вибору текстур «**Select Texture 2D**»

В інспекторі об'єктів натискаємо на кнопку «**Raise/Lower terrain**», вибираємо пензлик, встановлюємо розмір (**Brush Size**), інтенсивність (**Opacity**) (рис. 1.16).

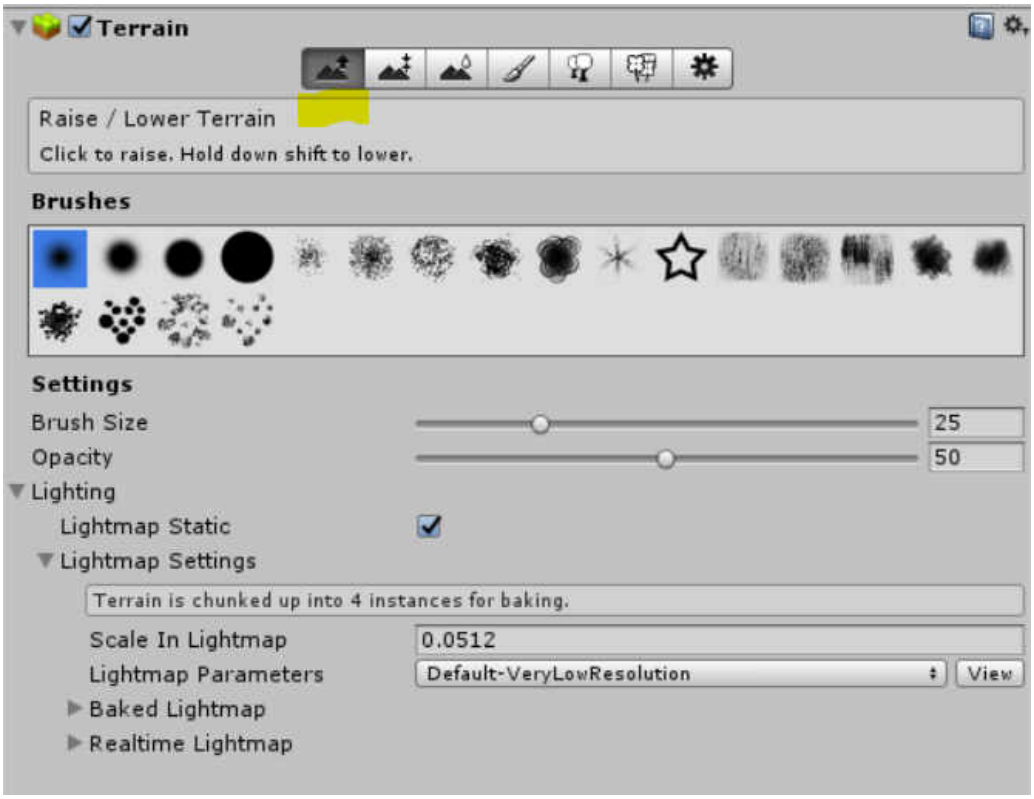

Рисунок 1.16 – Інспектор об'єктів. Кнопка «**Raise/Lower terrain**»

Натискаючи **ЛКМ** піднімаємо вершину, або натискаючи комбінацію **Shift + ЛКМ** опускаємо. Натискаючи кнопку **«Smooth Height»** прибираємо гострі кути (рис. 1.17).

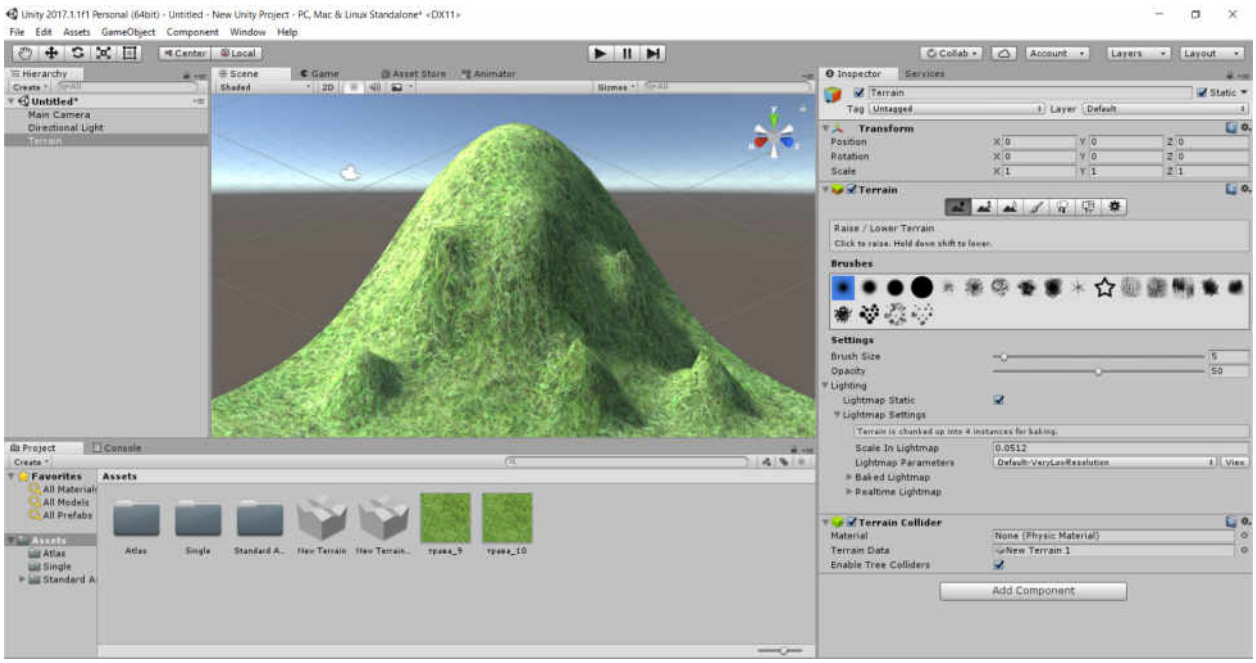

Рисунок 1.17 – Створення об'єкту **Terrain**

7. Робота з освітленням і камерою

Джерела світла використовуються для освітлення сцени та об'єктів.

В Unity розробнику доступні 5 типів джерел світла:

Directional Light. Найпростіше джерело світла, імітує сонячне світло. **Directional Light** є нескінченною множиною паралельних променів.

Point Light. Точкове джерело світла, тобто промені розходяться в різні боки з однієї точки.

Spot lights. Джерело світла, що світить з точки в деякому напрямку і висвітлює об'єкти тільки всередині конуса.

Area Light. Джерело світла, що має площу. Уявіть собі прямокутну панель, з якої виходить світло, це і буде **Area Light.**

Light Probe Group. Особливе джерело світла, що впливає виключно на динамічні об'єкти.

Для створення джерела світла виберемо меню **«GameObject→Light»** (рис.1.18).

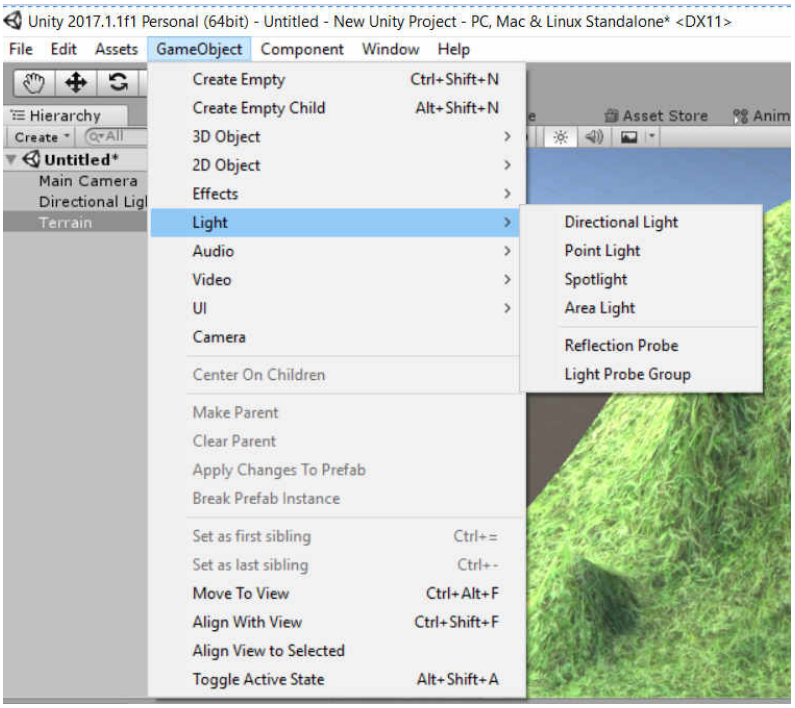

Рисунок 1.18 – Створення джерела світла

Спрямоване світло Directional Light.

Джерела спрямованого світла в основному використовуються для сонячного і місячного світла на сценах з відкритим простором. Джерела світла даного типу впливають на поверхні всіх об'єктів сцени. Також це найдешевше джерело світла для графічного процесора.

Для створення джерела світла **Directional Light** виберемо меню **«GameObject→Light→Directional Light»** (рис.1.19).

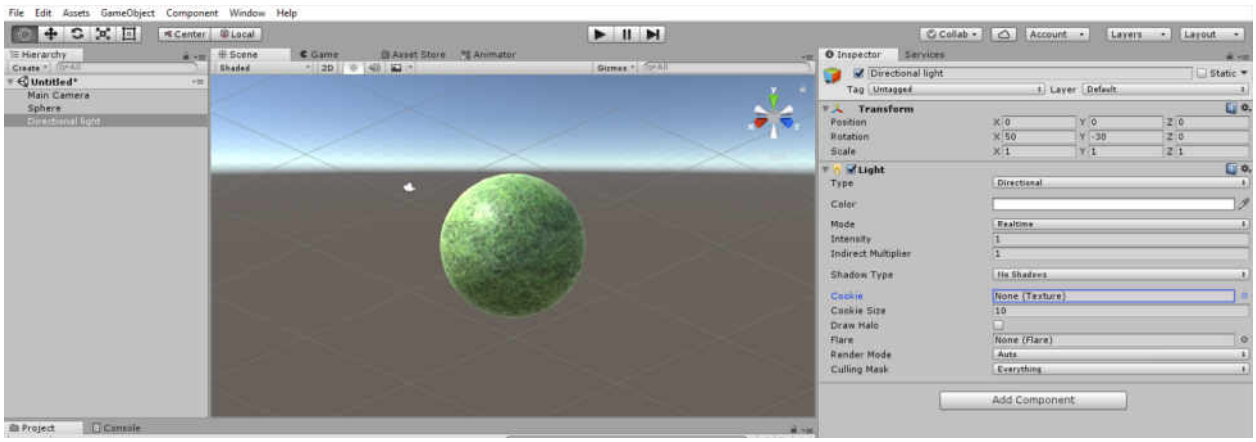

Рисунок 1.19 – Створення джерела світла **Directional Light**

Точкове освітлення (Point Lights)

Point light світить на всі боки з однієї точки. Це найпоширеніше джерело світла в іграх. Даний тип освітлення має середню ресурсовитратність для графічного процесора.

Для створення джерела світла **Point Lights** виберемо меню **«GameObject→Light→ Point Lights»** (рис.1.20).

В інспекторі на вкладці **«Lights»** встановимо параметр «Range=60», параметр «Color» змінимо на «Red», встановимо параметр «Intensity=50».

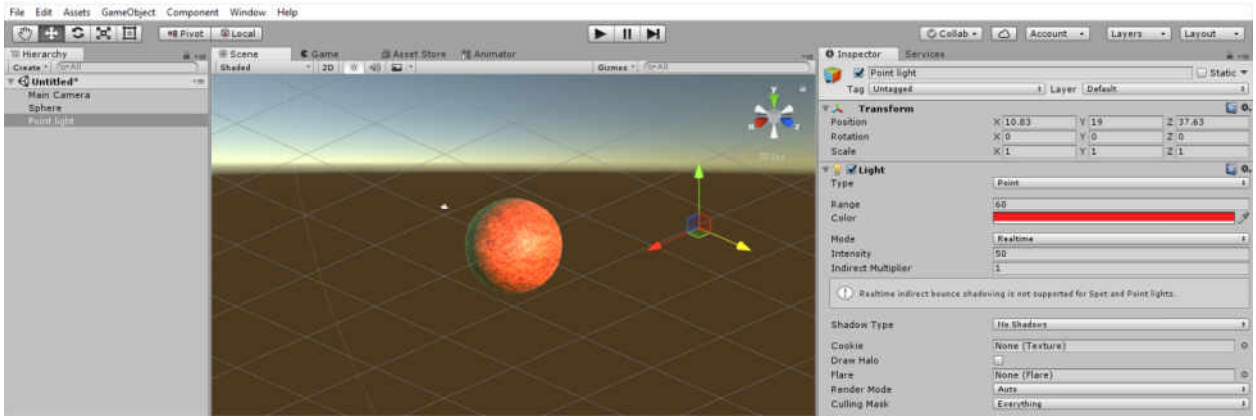

Рисунок 1.20 – Створення джерела світла **Point Lights**

Освітлення променем (Spot Lights)

Джерело світла **Spot Lights** світить в одному напрямку, всередині конуса. Джерела світла даного типу ідеально підходять для ручних ліхтариків, автомобільних фар або ліхтарних стовпів. Даний тип освітлення є ресурсозатратним.

Для створення джерела світла **Spot Lights** виберемо меню **«GameObject→Light→ Spot Lights»** (рис.1.21).

В інспекторі на вкладці **«Lights»** встановимо параметр «Range=60», параметр «Color» змінимо на «Red», встановимо параметр «Intensity=60».

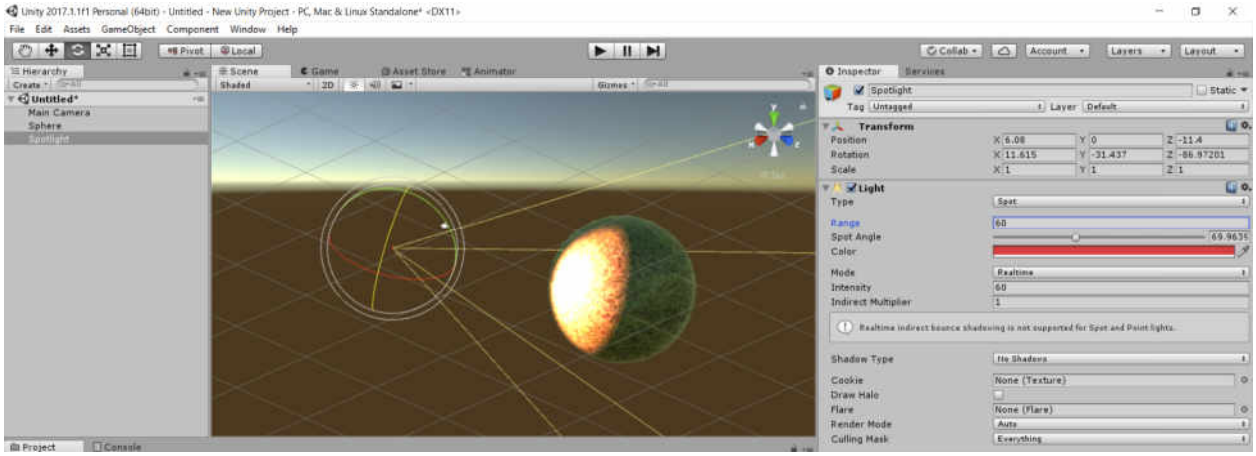

Рисунок 1.21 – Створення джерела світла **Spot Lights**

Плоске джерело світла (Area Lights)

Плоске джерело світла випромінює світло однієї зі сторін прямокутної області площини. Даний тип джерела світла впливає на об'єкти всередині області дії. Розмір прямокутника задається властивостями **«Width»** і **«Height»**. Світло випромінюється з усією поверхні прямокутної області. Оскільки **Area Lights** вимагає великих обчислень процесора, плоскі джерела світла недоступні під час гри. Тому можливе тільки запікання в світлові карти.

Для створення джерела світла **Area Lights** виберемо меню **«GameObject→Light→ Area Lights»** (рис.1.22).

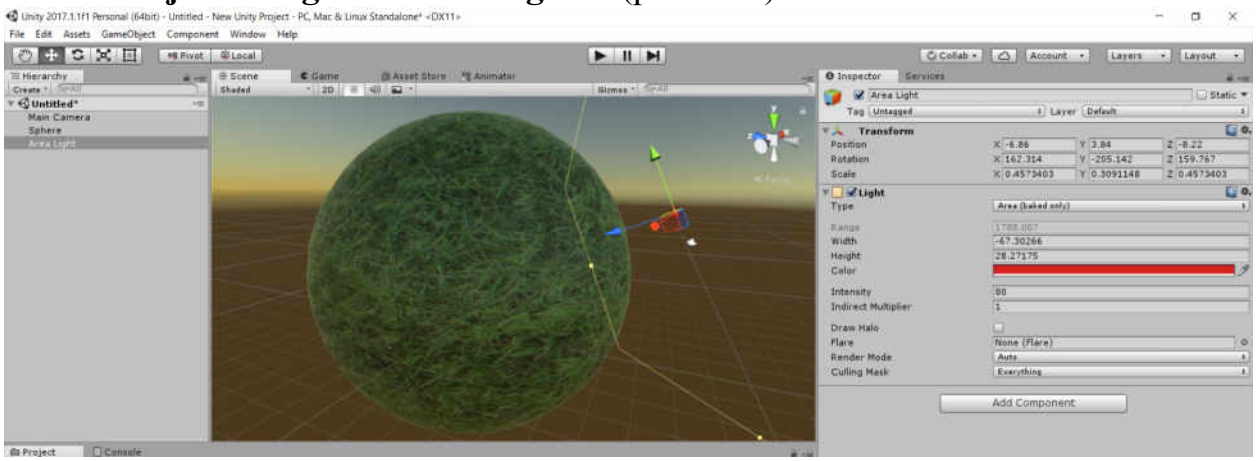

Рисунок 1.22 – Створення джерела світла **Area Lights**

Камери є пристроями, які захоплюють і відображають ігровий світ. Ви можете мати необмежену кількість камер в сцені та налаштувати рендеринг камерами в будь-якому порядку, на будь-якому місці екрану, або тільки в певних його частинах.

Для створення камери виберемо меню **«GameObject→Camera»** (рис.1.23).

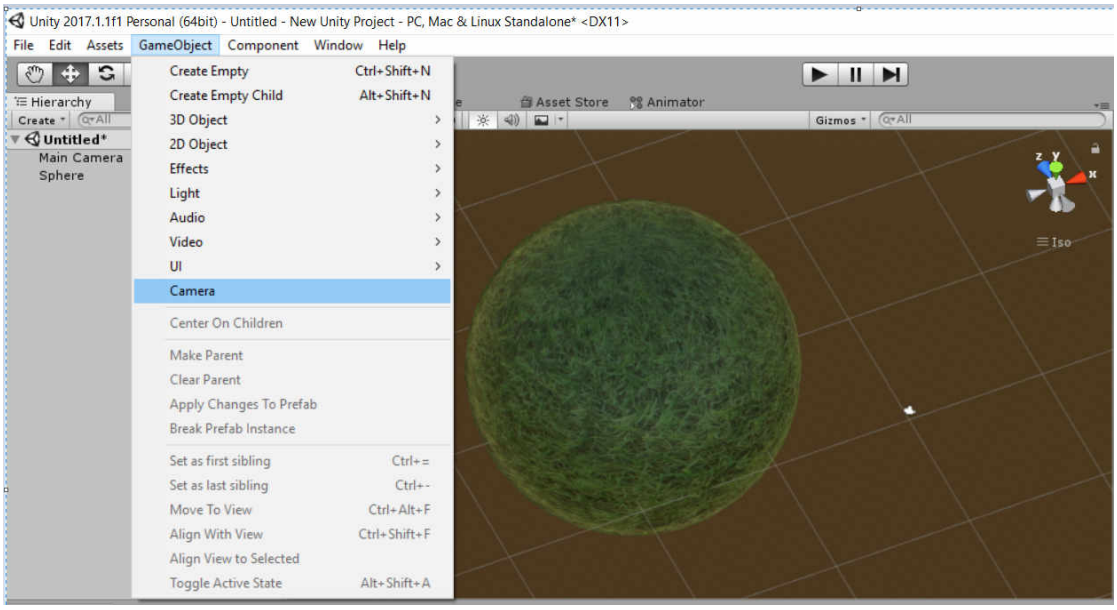

Рисунок 1.23 – Меню **«GameObject→Camera»**

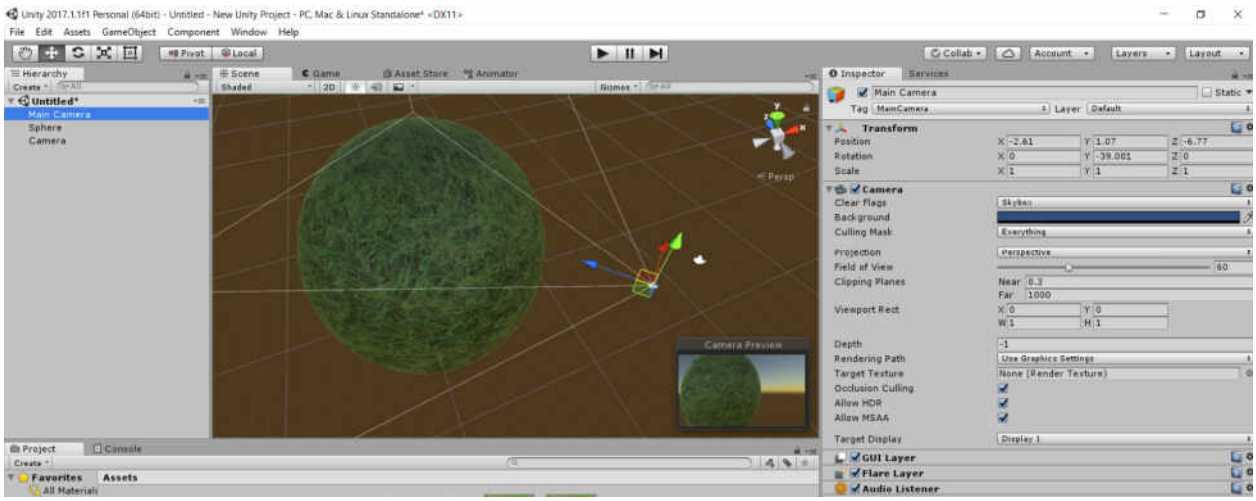

Рисунок 1.24 – Створення камери

Таблиця 2 – Властивості об'єкта «Camera»

Властивість	Функція
Clear Flags	Визначає, які частини екрану будуть очищені. Це зручно при використанні декількох камер для відтворення різних елементів гри.
Background	Колір, який застосовується для фону після відтворення всіх елементів, в разі відсутності скайбокса.
Culling Mask	Включення або виключення шарів об'єктів на рендер цією камерою. Призначення шарів об'єктів проводиться через Inspector.
Projection	Перемикає здатність камери симулювати перспективу. **Perspective** Камера буде рисувати об'єкти в перспективі. **Orthographic** Камера буде рисувати об'єкти рівномірно, без ефекту перспективи.
Size (коли обраний орографічний режим)	Розмір зони видимості камери для орографічного режиму.
Field of view (коли обраний режим перспективи)	Ширина кута огляду камери, вимірюється в градусах по локальній осі Y.
Clipping Planes	Дистанція, на якій камера починає і закінчує рендеринг. **Near** Найближча точка відносно камери, яка буде рисуватися. **Far** Дальня точка відносно камери, яка буде рисуватися.
Normalized View Port Rect	Чотири значення, що відображають те, в якій області екрану буде виведено зображення з камери, в екранних координатах (від 0 до 1). **X** Початкова позиція області по горизонталі виду камери, який буде рисуватися. **Y** Початкова позиція області по вертикалі, де вид камери буде рисуватися. **W (Ширина)** Ширина виду камери на екрані. **H (Висота)**

	Висота виду камери на екрані.
Depth	Позиція камери в черзі на прорисування. Камери з великим значенням будуть нарисовані поверх камер з меншим значенням.
Rendering Path	Опції для визначення методів рендеринга для камери. **Use Player Settings** Камера використовує метод візуалізації, встановлений в Player Settings. **Vertex Lit** Всі об'єкти, що рисуються цією камерою будуть рендеритьсяя як Vertex-Lit-об'єкти. **Forward** Всі об'єкти будуть рендеритьсяя з одним проходом на матеріал
HDR	Включення технології High Dynamic Range.

Завдання до лабораторної роботи №1

1. Створити новий проект Unity.
2. Здійснити налаштування інтерфейсу.
3. Здійснити імпорт ресурсів (матеріалів, текстур).
4. За допомогою 3D-об'єкту «Terrain» згенерувати ландшафт та додати текстуру ландшафту.
5. За допомогою меню «Game Object→ 3D Object→Sphere» створити 3D-об'єкт «Sphere» та прив'язати до нього 4 типи джерел світла (Directional Light, Point Light, Spot lights, Area Light).
6. За допомогою меню «Game Object→Camera» створити об'єкт «Camera» та прив'язати його до об'єкту «Sphere».
7. Здійснити рендеринг.

Контрольні запитання

- Назвіть базові графічні компоненти системи Unity 3D для роботи з освітленням та камерою.
- Перерахуйте основні елементи інтерфейсу системи Unity 3D.
- Назвіть базові компоненти системи Unity 3D для створення та редагування ландшафтів.
- Дайте визначення поняттю «рендеринг».

ЛАБОРАТОРНА РОБОТА № 2
ТЕМА: «КОМПОНЕНТИ UNITY 3D ДЛЯ РОБОТИ З ФІЗИКОЮ 3D-ОБ'ЄКТІВ»

Анотація

Лабораторна робота орієнтована на оволодіння студентами навичок роботи з основними компоненти, які використовуються для роботи з фізикою 3D - об'єктів.

Мета лабораторної роботи

Створення 3D - об'єктів та використання компонентів Rigidbodies, Colliders, Joints, Character Controllers для роботи з фізикою 3D - об'єктів.

Робота з властивістю «Is Kinematic» компонента Rigidbody, яка надає можливість виключення об'єкта з-під контролю фізичного движка, і дозволити переміщати його кінематично за допомогою відповідного скрипта.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти використовувати компоненти Rigidbodies, Colliders, Joints, Character Controllers та їх властивості для роботи з фізикою 3D - об'єктів, використовувати властивість «Is Kinematic» компонента Rigidbody для виключення об'єкта з-під контролю фізичного движка, і кінематичного переміщення об'єкта за допомогою відповідного скрипта.

Компоненти для роботи з фізикою 3D - об'єктів

Для того, щоб фізична поведінка об'єктів в ігровому додатку була реалістичною, об'єкт у грі потрібно правильно прискорити і задіяти зіткнення, гравітацію та інші сили. Вбудовані в Unity фізичні движки забезпечують вас компонентами для обробки симуляції фізики. За допомогою налаштування всього декількох параметрів, можна створити об'єкти, які поводять себе пасивно реалістично (тобто вони будуть переміщатися в результаті зіткнень і падінь, але не почнуть рухатися самі по собі). Керуючи фізикою за допомогою скриптів, ви можете придати об'єкту динаміку.

Unity має два окремих фізичних движка, один для 3D-фізики і один для 2D-фізики. Основні поняття ідентичні в обох движках (за винятком додаткового виміру в 3D), але вони реалізовані з різними компонентами. Так, наприклад, існує компонент Rigidbody для 3D фізики і аналогічний Rigidbody 2D для 2D фізики.

Розглянемо далі основні компоненти для роботи з фізикою 3D - об'єктів.

Rigidbodies (Тверде тіло)

Rigidbody - це основний компонент, що включає фізичну поведінку для об'єкта. Якщо додати компонент Rigidbody до об'єкту, то він негайно почне реагувати на гравітацію.

Створіть новий проект або нову сцену в готовому проекті.

Додайте 3D-об'єкт **Sphere** за допомогою меню «**GameObject→3D Object →Sphere**», 3D-об'єкт **Plane** за допомогою меню «**GameObject→3D Object→Plane**» і точкове джерело світла **Point Light** за допомогою меню «**GameObject→Light→ Point Light**» (рис.2.1).

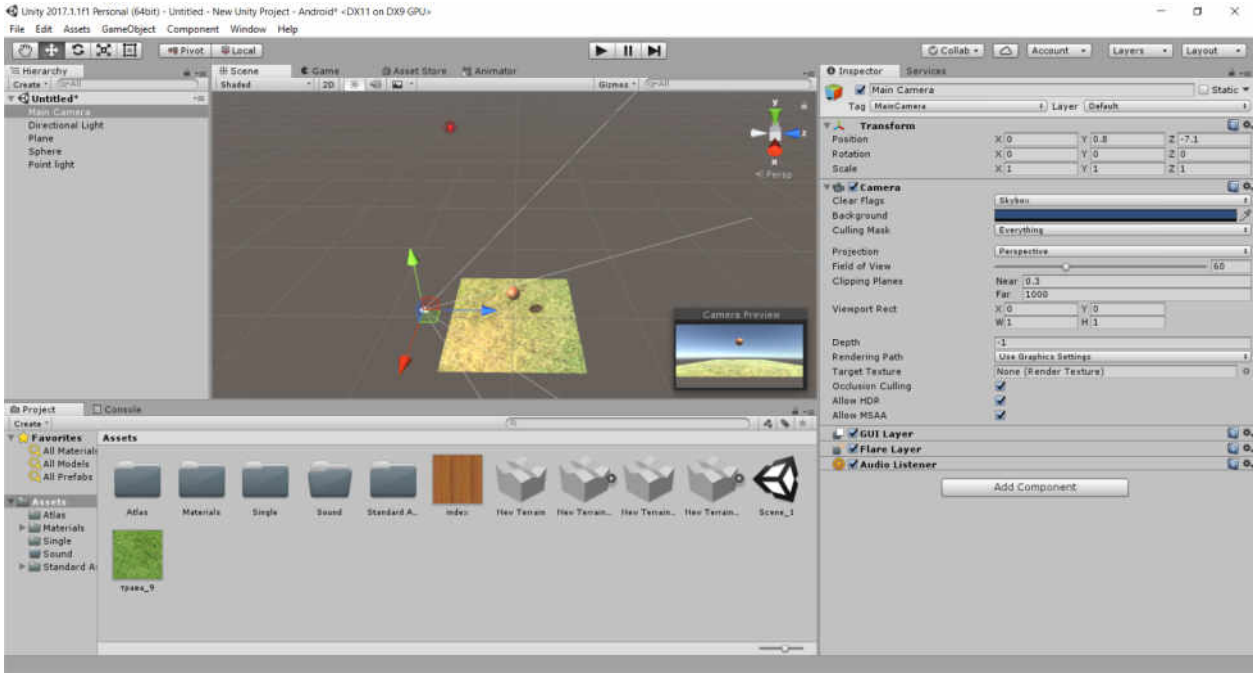

Рисунок 2.1 – Створення 3D-об'єктів

Виберіть 3D-об'єкт **Sphere** на панелі «**Hierarchy**». Після чого на панелі «**Inspector**» буде відкрито доступ до властивостей 3D-об'єкту **Sphere.** За допомогою кнопки «**Add Component→Physics→ Rigidbody**» додайте компонент **Rigidbody** до об'єкту **Sphere.** Доступ до властивостей компоненту **Rigidbody** також буде відкрито на панелі «**Inspector**» (рис.2.2).

Якщо натиснути на кнопку «**Play**», ви побачите як 3D-об'єкт **Sphere** падає на 3D-об'єкт **Plane** під дією сили тяжіння (рис.2.3).

Компонент **Rigidbody** має властивість «**Is Kinematic**», яка дозволяє вилучити об'єкт з-під контролю фізичного движка, і дозволити переміщати його кінематично за допомогою скрипта. Значення «**Is Kinematic**» можна

змінювати за допомогою коду, щоб ввімкнути або вимкнути фізику для об'єкта, але ця можливість вимагає додаткових ресурсів.

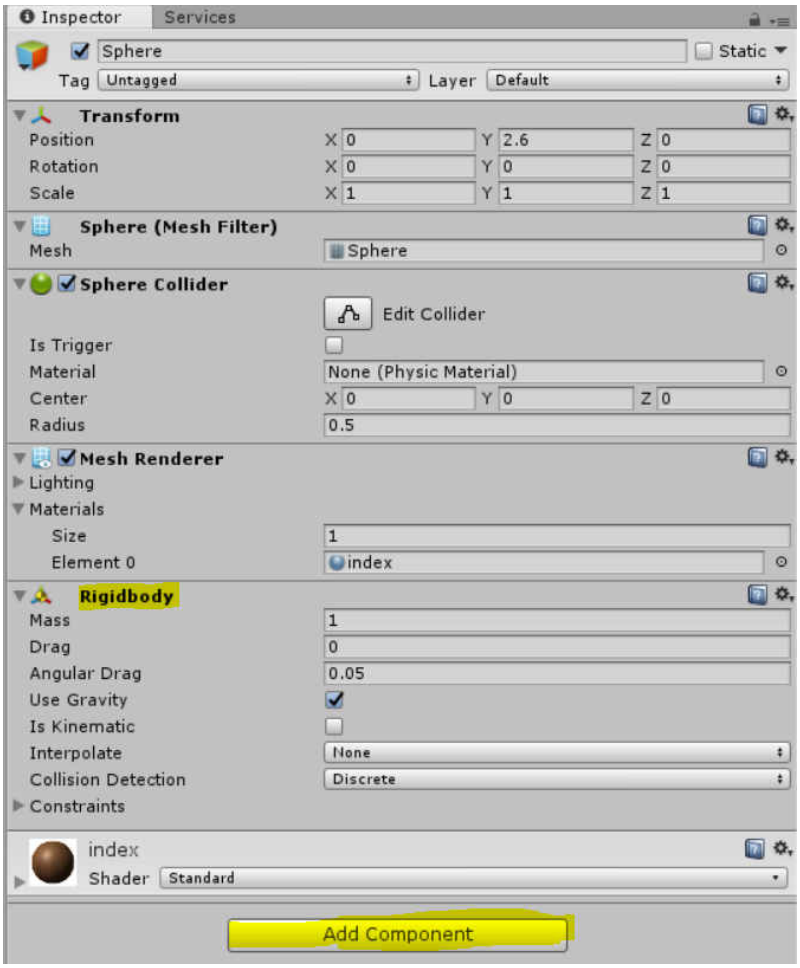

Рисунок 2.2 – Властивості компоненту **Rigidbody**

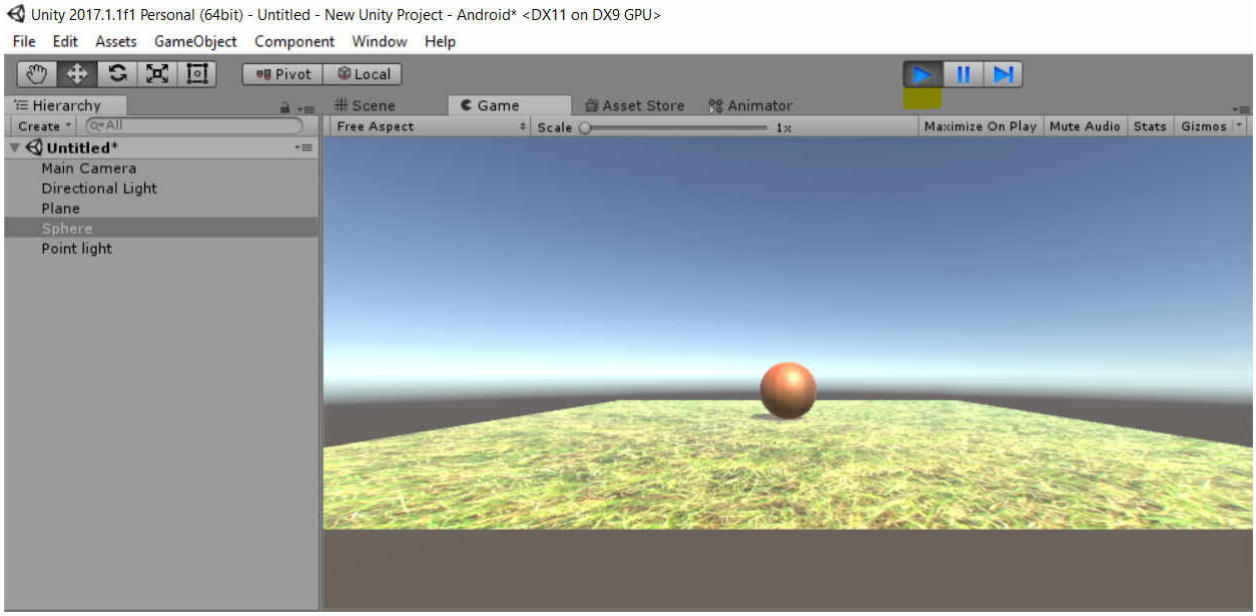

Рисунок 2.3 – Кнопка «**Play**»

Властивості компоненту **Rigidbody** подано у табл. 2.1.

Таблиця 2.1 - Властивості компоненту **Rigidbody**

Властивість	Функція
Mass	Маса об'єкта (за замовчуванням в кілограмах).
Drag	Повітряний опір, який діє на об'єкт поки він переміщується під впливом цих сил. 0 означає відсутність опору, а нескінченність (infinity) припиняє переміщення об'єкта.
Angular Drag	Повітряний опір, який діє на об'єкт поки він обертається під впливом сили обертання. 0 означає відсутність опору. Слід зазначити, що не можна зупинити обертання об'єкта шляхом установки його кутового опору (Angular Drag) в нескінченне (infinity) становище.
Use Gravity	При включенні на об'єкт діє гравітація.
Is Kinematic	При включенні вилучає об'єкт з-під контролю фізичного движка, і дозволяє переміщати його кінематично за допомогою скрипта.
Interpolate	Застосовується, якщо помічено тряску в переміщенні твердого тіла. - None Не застосовано жодної інтерполяції. - Interpolate Згладжування трансформації, що засноване на трансформації з попереднього кадру. - Extrapolate Згладжування трансформації, що засноване на приблизної трансформації наступного кадру.
Collision Detection	Використовується для запобігання проникнення об'єктів, які швидко рухаються крізь інші об'єкти без визначення зіткнень. - Discrete Для всіх колайдерів сцени необхідно використовувати Discreet-виявлення зіткнень. Також колайдери використовуватимуть Discreet-виявлення зіткнень при перевірці на зіткнення проти них. Значення Discreet встановлено за умовчанням. - Continuous

	Використовується для виявлення зіткнень проти динамічних колайдерів (з твердими тілами) і безперервне виявлення зіткнень проти статичних меш колайдерів (без твердих тіл). Тверді тіла, встановлені як Continuous Dynamic будуть використовувати Continuous-виявлення зіткнень при тестуванні на зіткнення проти них. Інші тверді тіла також будуть використовувати Continuous-виявлення зіткнень. Використовується для об'єктів, з якими потрібно зіткнутися за допомогою динамічного виявлення зіткнень. Все це дуже впливає на фізичну продуктивність, тому залиште дане значення встановленим в Discrete, якщо не відчуваєте проблем із зіткненням об'єктів, які швидко рухаються. - Continuous Dynamic Використовується безперервне і безперервне динамічне виявлення зіткнень проти об'єктів налаштованих на безперервне і безперервне динамічне зіткнення. Також буде використовуватися безперервне виявлення зіткнень проти статичних меш колайдерів (без твердих тіл). Для всіх інших колайдерів буде використовуватися Discreet-виявлення зіткнень. Використовується для об'єктів, які швидко рухаються.
Constraints	Обмеження руху твердого тіла - Freeze Position Вибірково зупиняє переміщення твердого тіла по осях X, Y і Z. - Freeze Rotation Вибірково зупиняє обертання твердого тіла по осях X, Y і Z.

Colliders (Колайдери)

Колайдери це наступний тип компонентів, які повинні бути додані до твердих тіл, щоб задіяти зіткнення. Якщо два твердих тіла врізаються один в одного, фізичний движок не буде прораховувати зіткнення, поки до обох об'єктів не буде додано колайдер. Тверді тіла, які не мають колайдерів будуть просто проходити крізь один одного при прорахунку зіткнень.

Давайте трохи змінимо фізичні властивості 3D-об'єкту **Sphere**. Знову виберіть 3D-об'єкт **Sphere**. На панелі «**Inspector**» за допомогою кнопки «**Add**

Component→Physics→ Box Collider» додайте компонент **Box Collider** до об'єкту **Sphere.** Доступ до властивостей компоненту **Box Collider** також буде відкрито на панелі «**Inspector**» (рис.2.4).

Box Collider базовий кубічний примітив зіткнень.

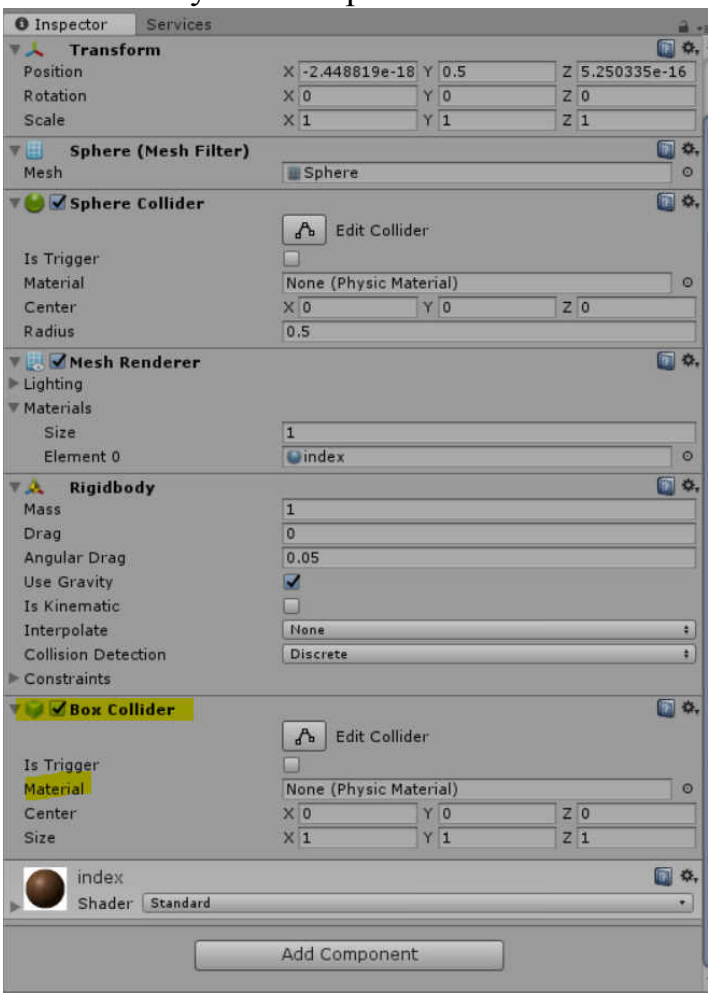

Рисунок 2.4 – Компоненту **Box Collider**

Властивості компоненту **Box Collider** подано у табл. 2.2.

Таблиця 2.2 - Властивості компоненту **Box Collider**

Властивість	Функція
Is Trigger	Колайдер використовується для запуску подій, і ігнорується фізичним движком.
Material	Визначає, як цей колайдер взаємодіє з іншими.
Center	Позиція колайдера.
Size	Розміри колайдера в напрямках X, Y, Z.

Одним з параметрів компонента **Box Collider** є **Material**, який за замовчуванням має значення **None** (Physics Material).

Для того, щоб створити новий фізичний **Material** скористайтеся меню «**Assets→Create→Physic Material**» (рис. 2.5).

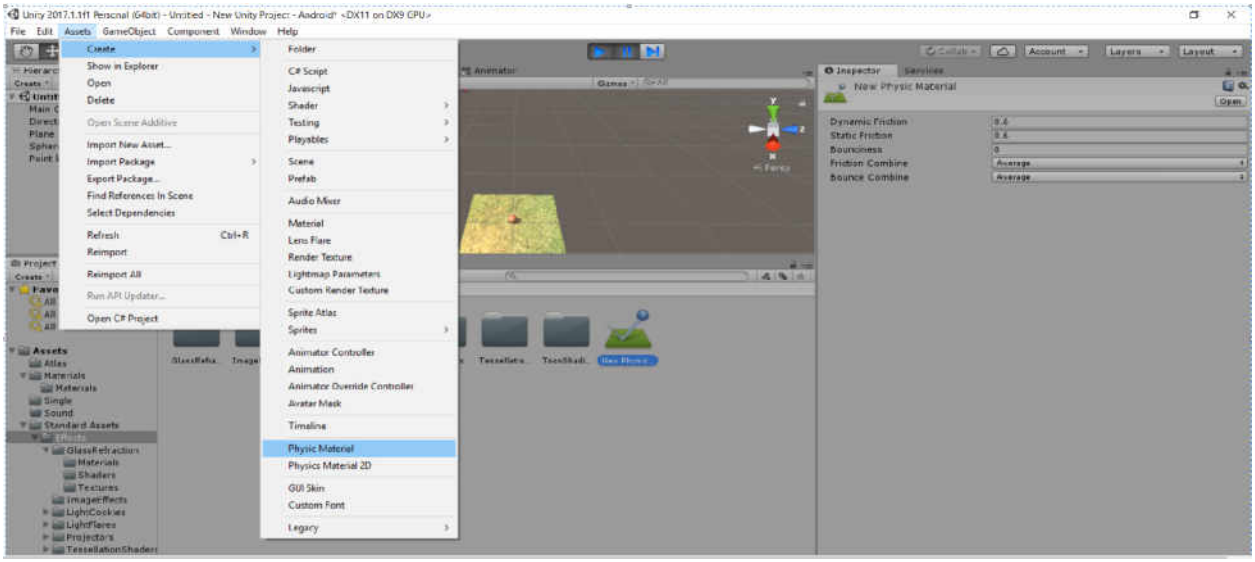

Рисунок 2.5 – Створення нового **Physics Material**

Властивості **Physics Material** подано у табл. 2.3.

Таблиця 2.3 - Властивості **Physics Material**

Властивість	Функція
Dynamic Friction	Тертя, що використовується під час руху. Зазвичай використовуються значення в діапазоні від 0 до 1. Значення, яке дорівнює 1 відповідає тертю як на льоду, в той час як значення, яке дорівнює 0 означає слабке тертя, в результаті якого об'єкту буде складно рухатися без впливу зовнішніх сил.
Static Friction	Тертя, що використовується коли об'єкт лежить на поверхні. Зазвичай використовуються значення в діапазоні від 0 до 1. Значення, яке дорівнює 0 означає відсутність тертя, в той час як значення, яке дорівнює 1 означатиме абсолютне тертя (тобто об'єктам по такій поверхні буде складно пересуватися).
Bounciness	Вказує наскільки пружною є поверхня. Значення , яке дорівнює 0 означає непружну поверхню. Значення, яке дорівнює 1 призведе до пружності, при якій об'єкт не буде втрачати початкову енергію.
Friction Combine	Вказує на те, як комбінується між собою тертя двох об'єктів.

	- Average Середнє значення - Minimum З двох значень використовується те, що менше. - Maximum З двох значень використовується те, що більше. - Multiply Значення множаться один на одне.
Bounce Combine	Вказує на те, як комбінується пружність двох об'єктів. Bounce Combine підтримує ті ж самі режими, що і Friction Combine режим.

Задайте значення **Physics Material,** як показано на рис. 2.6.

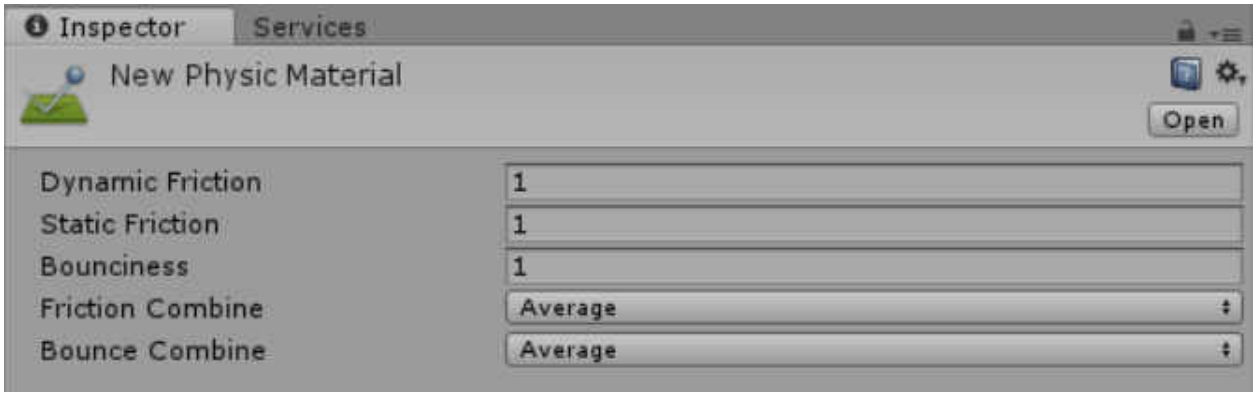

Рисунок 2.6 – Властивості **Physics Material**

Додайте щойно створений **Physics Material до** компоненту **Box Collider.**

Якщо натиснути на кнопку «**Play**», ви побачите як після падіння 3D-об'єкт **Sphere** почне відскакувати від поверхні (3D-об'єкт **Plane)**.

Fixed Joints (Нерухоме з'єднання)

Компонент **Fixed Joints** обмежує рух певного об'єкта, пов'язуючи його з іншим об'єктом.

Найчастіше цей компонент використовується, якщо в певний момент часу може знадобитися роз'єднати два об'єкти, або навпаки, з'єднати два об'єкти без необхідності зміни ієрархії.

Властивості **Fixed Joints** подано у табл. 2.4.

Таблиця 2.4 - Властивості **Fixed Joints**

Властивість	Функція
Connected Body	Необов'язкове посилання на інший об'єкт з Rigidbody, до якого приєднується поточний об'єкт. Якщо поле залишити порожнім, об'єкт приєднується до заданої точки в просторі.
Break Force	Сила, яку потрібно прикласти до об'єкта, щоб розірвати з'єднання.
Break Torque	Крутний момент, який необхідно прикласти до об'єкта, щоб розірвати з'єднання.
Enable Collision	Якщо включено Enable Collision, то два з'єднаних об'єкта будуть стикатися один з одним.
Enable Preprocessing	Вимкнення попередньої обробки допомагає стабілізувати неможливі конфігурації.

При створенні ігор іноді виникають випадки, коли потрібно, щоб об'єкти рухались разом (тимчасово або постійно). Компоненти **Fixed Joints** дозволяють спростити реалізацію подібних ситуацій, оскільки вам не доводиться змінювати положення об'єкта в ієрархії за допомогою скриптів.

Минус такого рішення в тому, що вам доведеться додати компоненти **Rigidbodies** на об'єкти, які потрібно з'єднати за допомогою **Fixed Joints.**

Додайте до 3D-об'єкту **Sphere** компонент **Fixed Joints**. На панелі «**Hierarchy**» виберіть 3D-об'єкт **Sphere**. На панелі «**Inspector**» за допомогою кнопки «**Add Component→Physics→Fixed Joint**» додайте компонент **Fixed Joint** до об'єкту **Sphere.** Доступ до властивостей компоненту **Fixed Joint** також буде відкрито на панелі «**Inspector**» (рис.2.7).

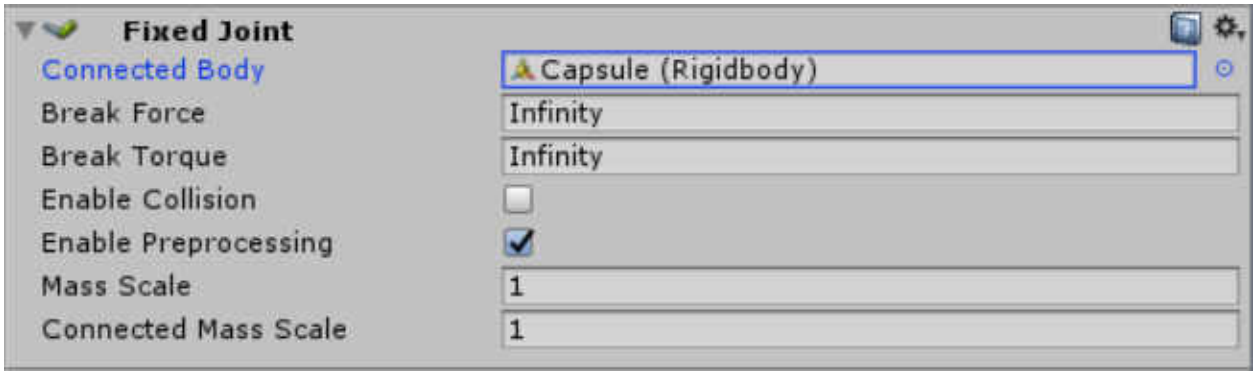

Рисунок 2.7 – Компонент **Fixed Joint**

З'єднайте 3D-об'єкт **Sphere** з будь-яким іншим компонентом на сцені за допомогою властивості **Connected Body** (в нашому випадку це 3D-об'єкт **Capsule).**

Компонент **Fixed Joints** обмежить рух 3D-об'єкту **Sphere**, з'єднавши його з іншим 3D-об'єктом **Capsule** .

4.4 Spring Joint (Пружне з'єднання)

Spring Joint з'єднує пружиною два Rigidbody-об'єкта.

Властивості **Spring Joint** подано у табл. 2.5.

Таблиця 2.5 - Властивості **Spring Joint**

Властивість	Функція
Connected Body	Об'єкт з пружинним з'єднанням з'єднаний з іншим об'єктом. Якщо об'єкту не призначено Connected Body, то пружина буде з'єднана з фіксованою точкою в просторі.
Anchor	Точка в локальному просторі об'єкта, на якій знаходиться Joint.
Auto Configure Connected Anchor	Unity автоматично обчислює позицію підключеної опорної точки?
Connected Anchor	Точка в локальному просторі з'єднаного об'єкту, на якій знаходиться Joint.
Spring	Сила пружини.
Damper	Значення, при якому пружину скорочено, якщо вона є активною.
Min Distance	Нижня межа діапазону відстані, над якою пружина не буде застосовувати будь-яку силу.
Max Distance	Верхня межа діапазону відстані, над якою пружина не буде застосовувати будь-яку силу.
Tolerance	Змінює толерантність до помилок.
Break Force	Сила, яку слід прикласти до Joint, щоб розбити його.
Break Torque	Крутний момент, який треба прикласти до Joint, щоб розбити його.
Enable Collision	Виявлення колізій (зіткнень).
Enable Preprocessing	Вимкнення попередньої обробки допомагає стабілізувати неможливі конфігурації.

Character Controllers (Контролери персонажа)

Компонент **Character Controller** використовується для управління від третьої або першої особи, де не потрібна фізика **Rigidbody**.

Цей компонент дає персонажу простий колайдер в формі капсули, який завжди знаходиться у вертикальному положенні.

У компонента Character Controller є свої особливі функції для призначення швидкості і напряму об'єкта (табл.2.6).

Таблиця 2.6 - Властивості **Character Controllers**

Властивість	Функція
Slope Limit	Обмежує можливість колайдера підійматися по схилах, значення яких дорівнює або менше ніж вказане в Slope Limit.
Step Offset	Персонаж ступить на поверхню, тільки якщо вона ближче до землі, ніж задане в Step Offset значення.
Skin width	Два колайдери можуть перетнутися один з одним на глибину, яка дорівнює значенню Skin Width. Найкращим варіантом є встановлення цього значення рівним 10% від радіуса.
Min Move Distance	Якщо персонаж спробує зрушити нижче зазначеної величини, то він не зрушить. У більшості ситуацій це значення варто залишити рівним нулю (0).
Center	Зрушення колайдера в світовому просторі без впливу на те, як обертається персонаж.
Radius	Значення радіуса колайдера.
Height	Висота колайдера Capsule Collider персонажа. Зміна значення Height розтягне колайдер уздовж осі X в обидва напрямки.

Додайте 3D-об'єкт **Capsule** за допомогою меню «**GameObject→3D Object → Capsule**», 3D-об'єкт **Plane** за допомогою меню «**GameObject→3D Object→Plane**».

На панелі «**Inspector**» ви побачите, що об'єкту **Capsule** призначено капсульний колайдер (рис.2.8).

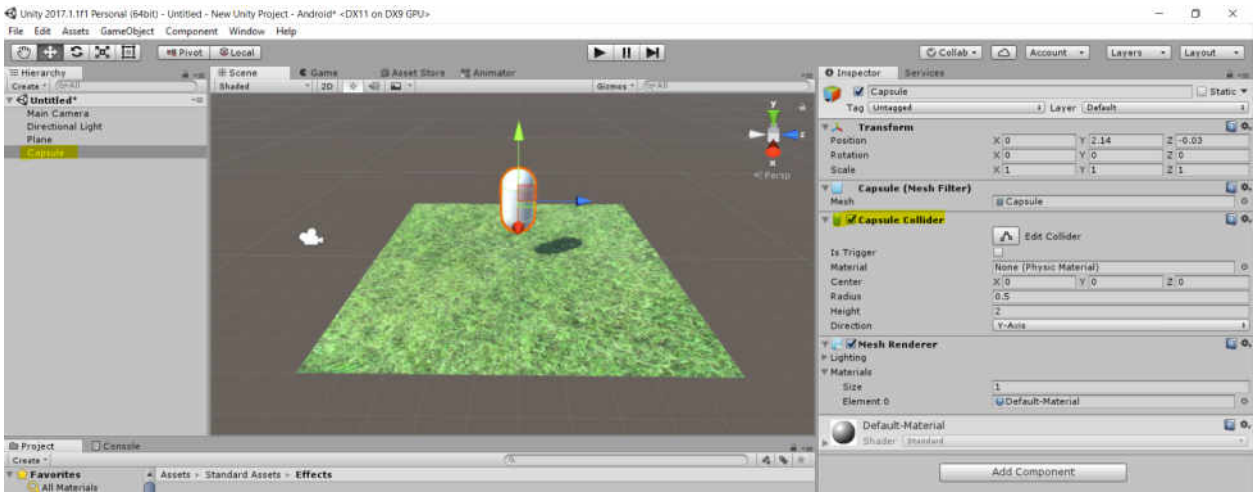

Рисунок 2.8 – 3D-об'єкт **Capsule**

Капсульний колайдер необхідно видалити. Для цього натисніть на значок із зображенням шестерні праворуч від імені компонента, як показано на рис. 2.9. Відкриється меню, в якому ви знайдете команду **Remove Component.**

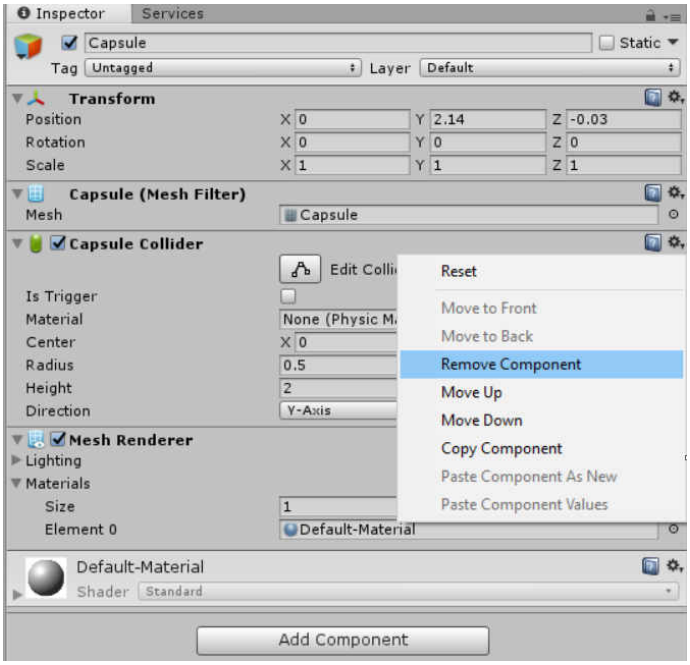

Рисунок 2.9 – Команда **Remove Component**

Замість капсульного колайдера призначимо об'єкту компонент **Character Controller**. На панелі «**Inspector**» ви знайдете кнопку «**Add Component**», за допомогою якої можна відкрити меню з переліком типів доступних компонентів. У розділі **Physics** знаходиться потрібний нам компонент **Character Controller.** Призначимо його 3D-об'єкту **Capsule** (рис.2.10).

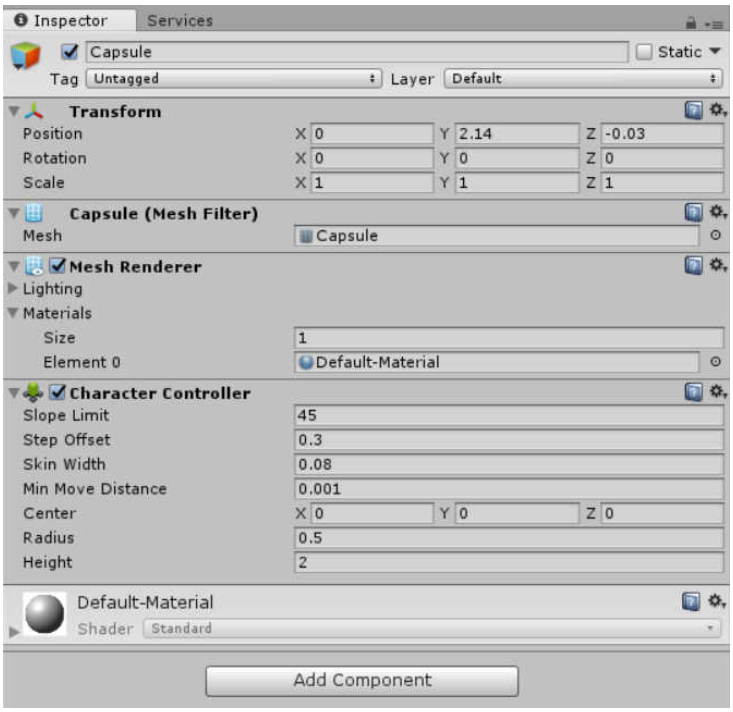

Рисунок 2.10 – Компонент **Character Controller**

Компонент **Character Controller** забезпечить природне переміщення 3D-об'єкту **Capsule.**

Для того, щоб переміщати 3D-об'єкт, який володіє компонентом **Characler Controller**, необхідно створити C# -скрипт в даному випадку з ім'ям **ExampleClass.**

На панелі **Project** в папці **Assets** за допомогою контекстного меню **«Create→C# Script»** створіть файл C# -скрипта з ім'ям **ExampleClass.**

Приклад C# -скрипта для переміщення 3D-об'єкту подано у лістингу 2.1.

За допомогою кнопки «**Open**» на панелі «**Inspector**» завантажте Visual Studio та відредагуйте C# -скрипт **ExampleClass** згідно лістингу 2.1.

Додати створений скрипт до 3D-об'єкту з компонентом **Characler Controller** можна шляхом перетягування скрипта на об'єкт в сцені.

Лістинг 2.1 Переміщення компоненту Character Controller

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public float speed = 6.0F;
    public float jumpSpeed = 8.0F;
    public float gravity = 20.0F;
    private Vector3 moveDirection = Vector3.zero;
    void Update() {
```

```
        CharacterController controller =
GetComponent<CharacterController>();
        if (controller.isGrounded) {
            moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0,
Input.GetAxis("Vertical"));
            moveDirection = transform.TransformDirection(moveDirection);
            moveDirection *= speed;
            if (Input.GetButton("Jump"))
                moveDirection.y = jumpSpeed;

        }
        moveDirection.y -= gravity * Time.deltaTime;
        controller.Move(moveDirection * Time.deltaTime);
    }
}
```

Натисніть на кнопку «**Play**», щоб побачити результат роботи C# - скрипта. Переміщати 3D-об'єкт **Capsule** можна за допомогою стрілок.

Завдання до лабораторної роботи №2

1. Створіть новий проект Unity.
2. Додайте 3D-об'єкт **Sphere** за допомогою меню «**GameObject→3D Object →Sphere**», 3D-об'єкт **Plane** за допомогою меню «**GameObject→3D Object→Plane**» і точкове джерело світла **Point Light** за допомогою меню «**GameObject→Light→ Point Light**».
3. Додайте компонент **Rigidbody** до об'єкту **Sphere.**
4. Натисніть на кнопку «**Play**», щоб побачити як 3D-об'єкт **Sphere** падає на 3D-об'єкт **Plane** під дією сили тяжіння.
5. За допомогою кнопки «**Add Component→Physics→ Box Collider**» додайте компонент **Box Collider** до об'єкту **Sphere.**
6. Створіть новий фізичний **Material** за допомогою меню «**Assets→Create→Physic Material**». Задайте всі необхідні властивості фізичного матеріалу.
7. Додайте створений **Physics Material** до компоненту **Box Collider.**
8. Натисніть на кнопку «**Play**», щоб побачити як після падіння 3D-об'єкт **Sphere** почне відскакувати від поверхні (3D-об'єкт **Plane)**.
9. Створіть новий проект Unity.
10. Додайте 3D-об'єкт **Capsule** за допомогою меню «**GameObject→3D Object → Capsule**», 3D-об'єкт **Plane** за допомогою меню «**GameObject→3D Object→Plane**».

11. Додайте до 3D-об'єкт **Capsule** компонент **Character Controller.**
12. Створіть C# -скрипт для переміщення 3D-об'єкту **Capsule.**
13. Натисніть на кнопку «**Play**», щоб побачити результат роботи C# - скрипта.

Контрольні запитання

- Назвіть етапи створення 3D - об'єктів в системі Unity 3D.
- Для чого у компонента Rigidbody використовується властивість «Is Kinematic»?
- Назвіть тип нефізичного руху, який здійснюється за допомогою коду.
- Який компонент дозволяє здійснювати обертання 3D - об'єкта навколо заданої точки та осі?

ЛАБОРАТОРНА РОБОТА № 3
ТЕМА: «РОБОТА В ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБКИ MONODEVELOP»

Анотація

Лабораторна робота орієнтована на оволодіння студентами навичок роботи в інтегрованому середовищі розробки MonoDevelop.

Мета лабораторної роботи

Знайомство з інтегрованим середовищем розробки MonoDevelop, налаштування MonoDevelop, редагування вихідного коду в MonoDevelop.

Очікувані результати

У разі успішного виконання лабораторної роботи студент буде вміти створювати, редагувати скриптові файли з використанням середовища розробки MonoDevelop.

Налаштування MonoDevelop

MonoDevelop - це інтегроване середовище розробки (IDE), що поставляється разом з **Unity**. IDE поєднує в собі функції текстового редактора з додатковими можливостями для налагодження і виконання інших завдань з управління ігровими проектами.

MonoDevelop встановлюється за умовчанням разом з **Unity**. Під час установки **Unity**, ви можете скасувати установку **MonoDevelop**.

Переконайтеся, що **MonoDevelop** встановлений в якості зовнішнього редактора скриптів в Preferences за допомогою меню «**Edit→ Preferences»**, а потім виберіть вкладку **External Tools**. Якщо в опції **External Script Editor** вибрана опція **MonoDevelop (built-in)**, **Unity** запустить **IDE MonoDevelop** і буде використовувати його в якості редактора за замовчуванням для всіх скриптових файлів.

Слід зауважити, що в опції **External Script Editor** в якості редактора для всіх скриптових файлів може бути вибране середовище розробки **Visual Studio.**

Перш ніж почати налагодження коду в **MonoDevelop**, вам спершу слід перевірити, що в **Preferences**, на панелі **External Tools** включена опція **Editor Attaching** (рис.3.1).

Переконайтеся, що в **BuildSettings** цільової платформи (меню: **File→ Build Settings**) включені опції **Development Build** і **Script Debugging** (рис.3.2).