Щоб відкрити вікно анімації для створення ключових кадрів анімації скористайтеся меню **Window → Animation** (або натисніть комбінацію клавіш **Ctrl + 6** на клавіатурі).

Для створення ключових кадрів анімації об'єкта гри виберіть об'єкт на сцені або в панелі ієрархії. Потім натисніть на кнопку **Add Property** у вікні анімації. Для створення анімованої літаючої камери виберіть камеру сцени, а потім натисніть **Add Property** у вікні анімації. Далі **Unity** виведе діалогове вікно для зазначення місця розташування в папці проекту даних анімації (кліпу анімації), завдання його імені і підтвердження. Задайте ім'я анімації **Animation.anim**.

Після цього **Unity** створить два активи: актив анімаційного кліпу (який містить дані всіх ключові кадрів) і актив контролера анімації (**Animation Controller**), який є прив'язкою до системи **Mecanim**.

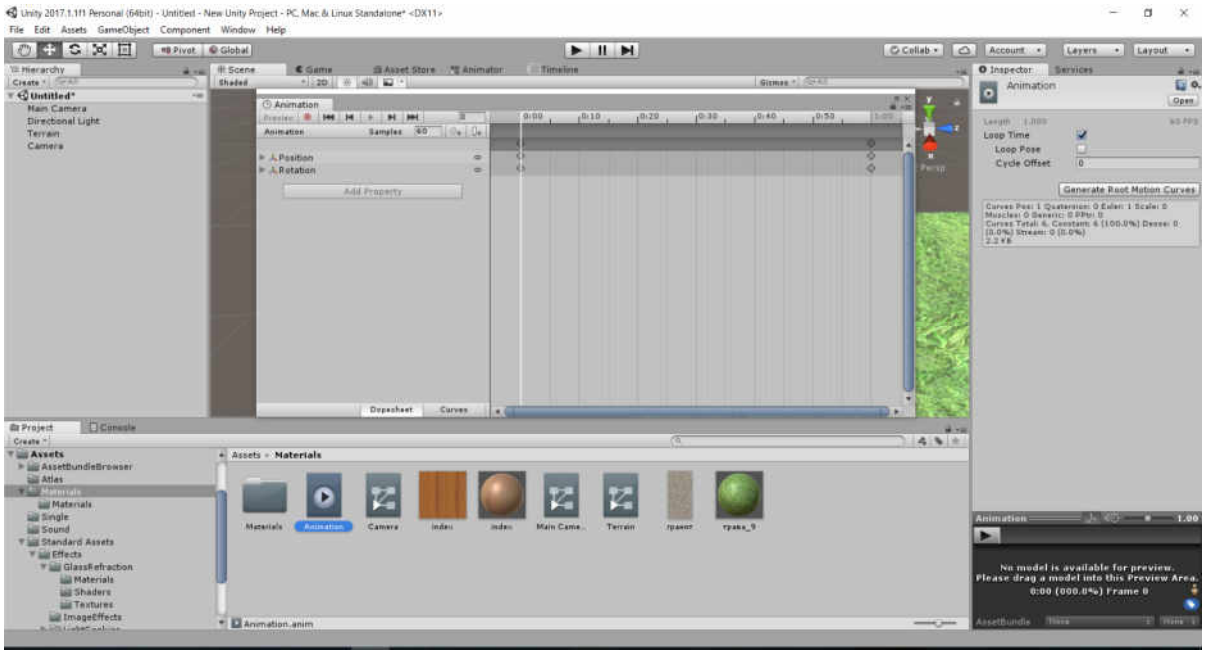

Рисунок 5.2 – Створення кліпу анімації **Animation.anim**

Щоб анімувати політ камери, натисніть на кнопку **Add Property** у вікні анімації і з контекстного меню виберіть канали для анімації. Зокрема, натисніть кнопку «+» і перейдіть до **Transform**. Потім ви можете перейти або до каналу положення (**Position**), або до каналу обертання (**Rotation**). При цьому у вікні анімації будуть додані два канали (положення і обертання), які тепер можуть бути анімовані.

Канал - це просто властивість, значення яке ми можемо прив'язати до ключових кадрів.

За замовчуванням для обох каналів положення і повороту автоматично генеруються два ключових кадра: один кадр на початку анімації (час: 0 секунд) і один в кінці (час: 1 секунда).

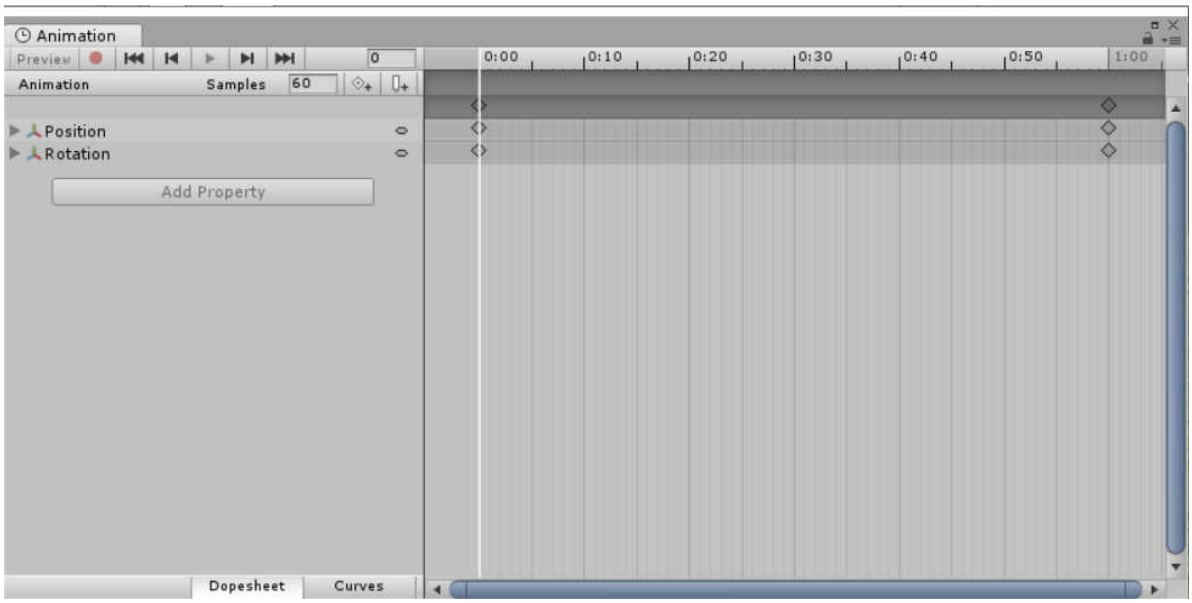

Рисунок 5.3 – Створення каналів анімації

Можна вручну створити ключовий кадр для запису стану будь-якого каналу (або всіх каналів) для камери. Для цього необхідно перетягнути ромбовидні значки ключових кадрів на шкалі часу, змінюючи для них час. Також можна продублювати їх за допомогою комбінації клавіш **Ctrl + C** і **Ctrl + V** і видалити вибрані ключові кадри за допомогою клавіш **Backspace** або **Delete**.

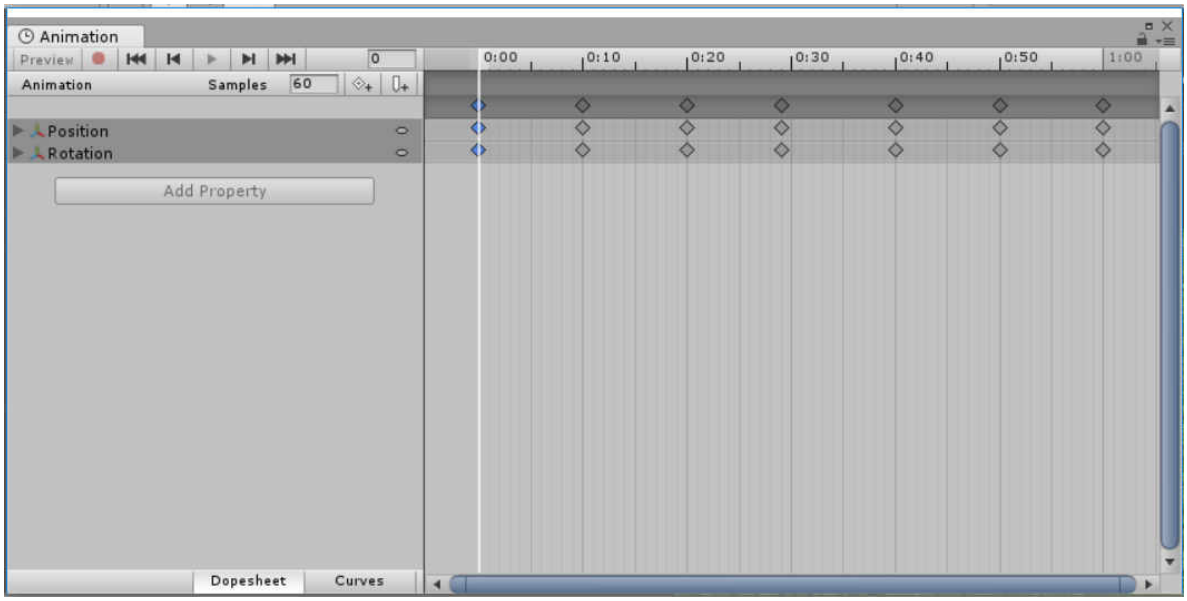

Рисунок 5.4 – Створення ключових кадрів анімації

В Unity існує можливість генерувати ключові кадри автоматично, ґрунтуючись на рухах об'єктів в сцені.

Для цього необхідно помістити повзунок анімації у вікні анімації в положення 0, а потім повернути об'єкт камери в сцені в її бажане вихідне положення і орієнтацію. Unity автоматично запише положення і поворот

камери для каналів на обраний вами момент часу. Повторіть цей процес генерації ключів кадрів, що визначають положення і поворот для камери, для кожного з моментів часу: 0:10, 0:20, 0:30, 0:40, 0:50, 1:00.

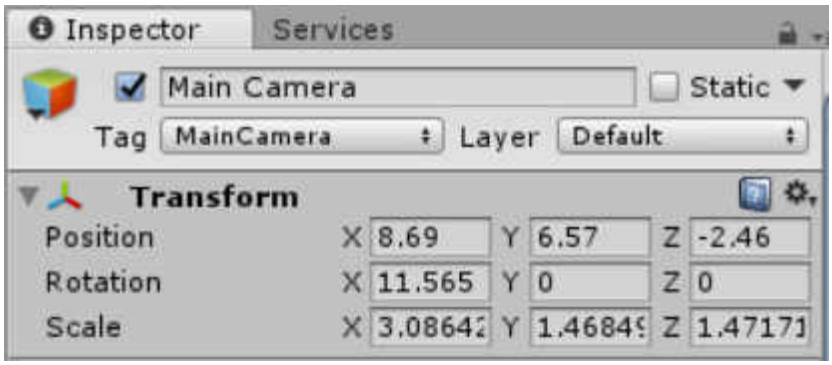

Рисунок 5.5 – Положення і поворот камери для каналів на обраний момент часу

Unity має широкий діапазон попередньо підготовлених систем частинок для створення найбільш поширених ефектів, таких як вибух, вогонь, пар або дим. Для отримання доступу до цих ефектів їх необхідно імпортувати до Unity-проекту за допомогою меню **Assets →Import Package→ Particle Systems**.

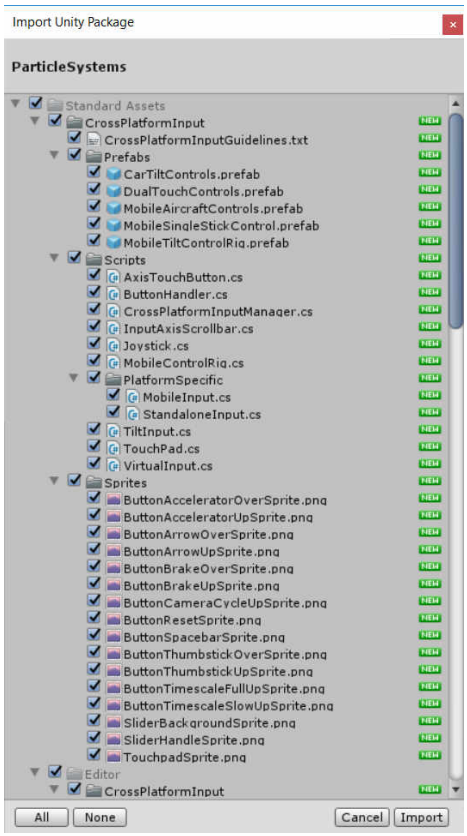

Рисунок 5.6 – Діапазон попередньо підготовлених систем частинок

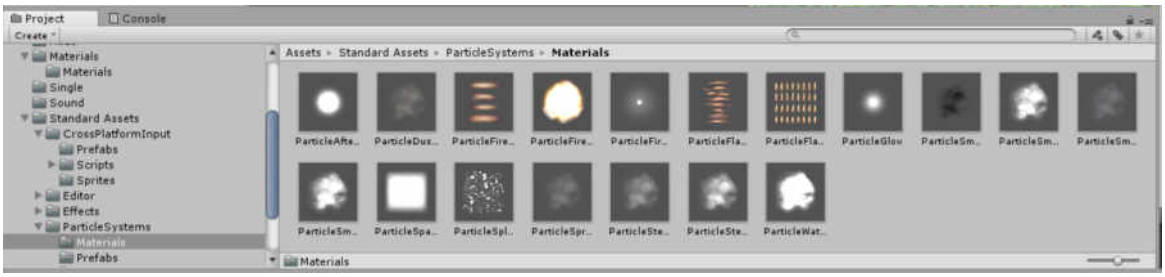

Рисунок 5.7 – Імпорт найбільш поширених ефектів

Всі системи частинок мають три спільні риси. По-перше, це наявність емітера (**emitter**), який, випускає частинки. По-друге, самі частинки - дрібні предмети, які генеруються емітером. По-третє, частинки мають термін служби – період часу, протягом якого вони існують, а також траєкторію або швидкість, яка визначає, що станеться з частинками протягом їх терміну служби (як вони будуть рухатися, як швидко переміщатися і т.д.).

Для роботи з частинками в Unity використовується система частинок Сюрікен (**Shuriken particle system**).

Створимо нову систему частинок в ігровій сцені. Для цього скористаємося меню **Game Object →Effects→ Particle System.**

Після вибору меню система частинок автоматично почне функціонувати в вікні сцени.

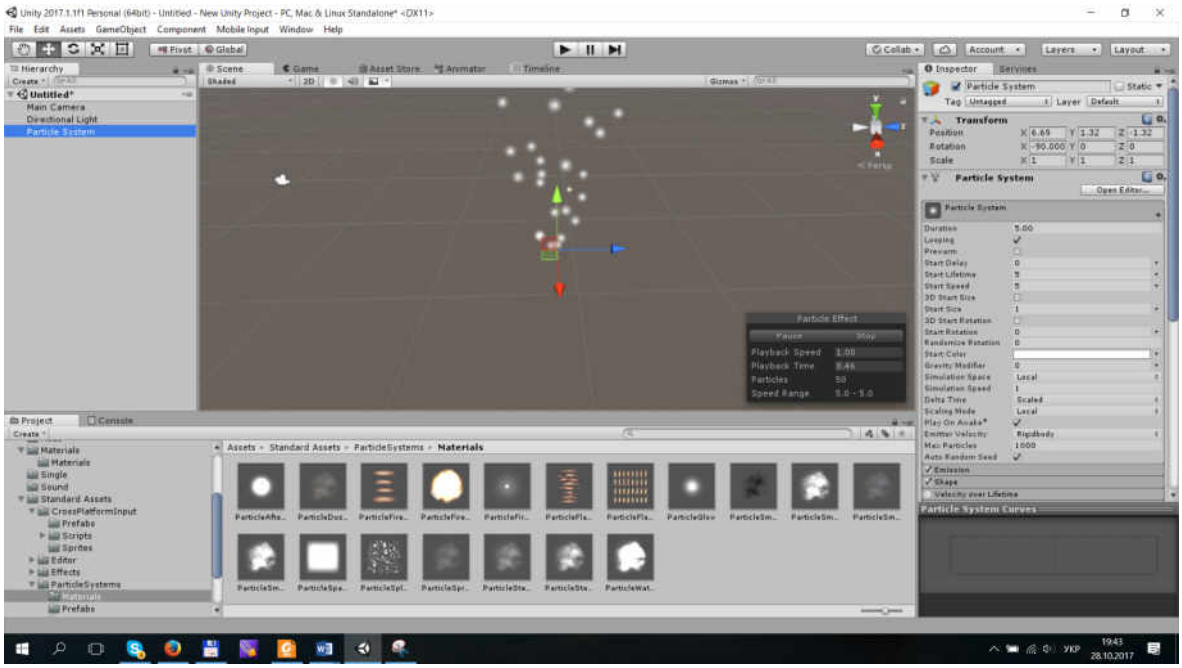

Рисунок 5.8 –Автоматичне функціонування системи частинок

Властивості системи частинок доступні на панелі «**Inspector**».

Рисунок 5.9 – Властивості системи частинок

Компонент **Particle System** має безліч налаштувань і для зручності, інспектор впорядковує їх у секції або «модулі». Кожна секція може бути розгорнута і згорнута при натисканні на заголовок. З лівого боку розташовані чекбокси, які можна відзначати для включення або відключення тих чи інших налаштувань. Наприклад, якщо ви не хочете змінювати розмір часток під час їх випромінювання, ви можете просто зняти галочку з параметра «**Size over lifetime**».

Крім модулів, в інспекторі є ще кілька інших елементів управління. Кнопка **Open Editor** відображає опції в окремому вікні редактора, яке також дозволяє редагувати одночасно кілька систем. А чекбокс **Resimulate**

відповідає за те, будуть чи не будуть застосовуватися зміни властивостей до щойно створених системою частинок. Кнопка **Wireframe** відображає контури меш-об'єктів, щоб показати частинки, які знаходяться за ними.

Значення поля тривалості **Duration** визначає загальну тривалість анімації системи частинок, перш ніж вона почнеться спочатку і зациклиться.

Значення поля початкового розміру **Start Size** управляє розміром згенерованих частинок.

Кожна система частинок має **емітер** – місце відображення частинок, які генеруються. Сам емітер має форму. Ця форма може бути площиною, точкою, кубом, кулею або звичайним мешем, і вона є поверхнею або тілом, всередині якого генеруються частинки.

Для управління формою емітера використовується закладка **Shape** в **Shuriken particle system.**

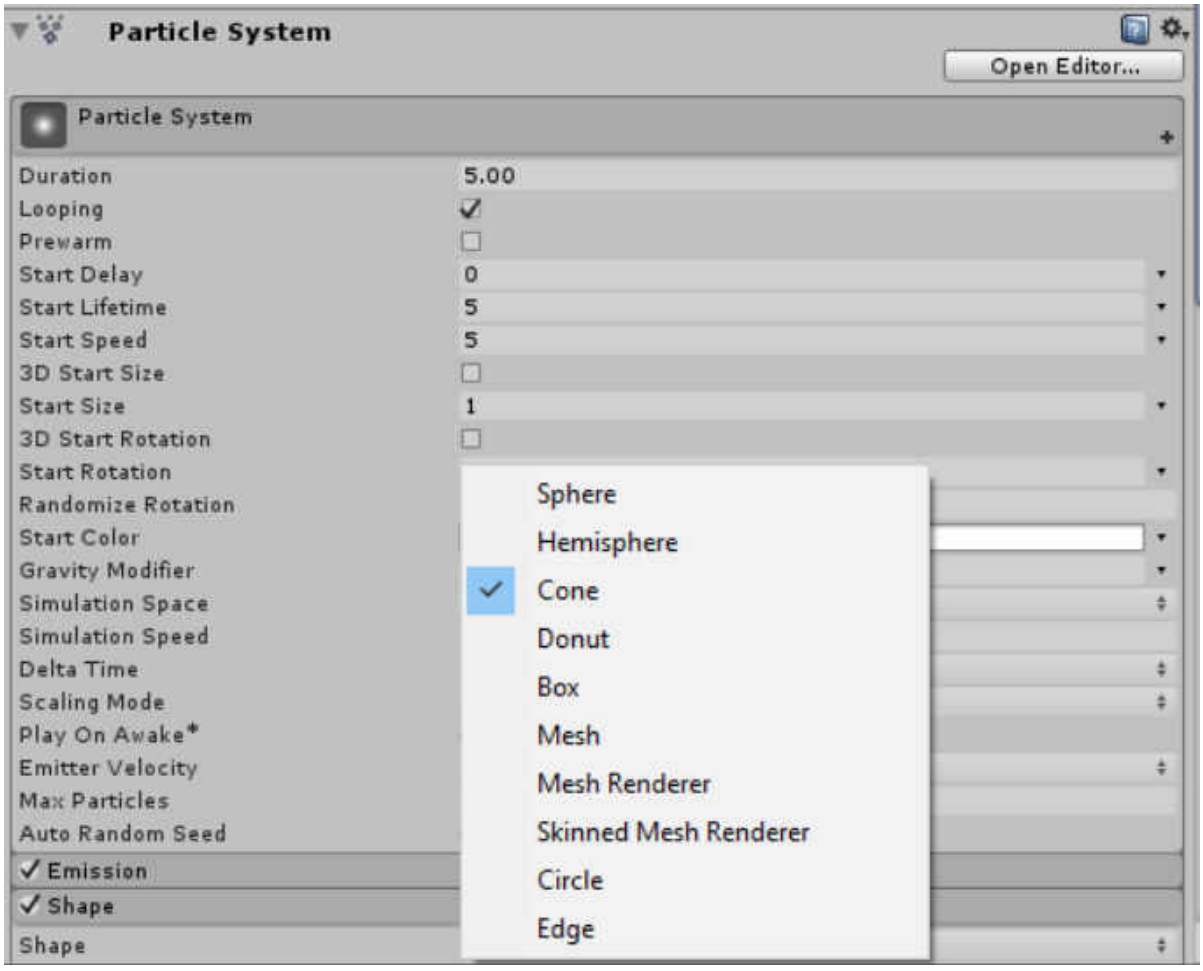

Рисунок 5.10 – Закладка **Shape**

За частоту або швидкість генерації нових частинок з емітера в секунду відповідає секція **Emission.**

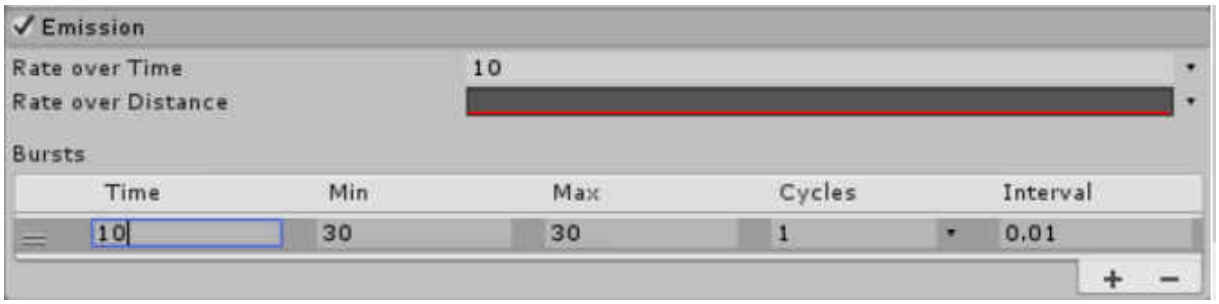

Рисунок 5.11 – Секція **Emission**

Параметри, що керують зовнішнім виглядом частинок, знаходяться в секції **Renderer.**

Поле **Render Mode** визначає, як візуалізуються частинки. Режимів всього два: режим білборда **Billboard** і режим мешів **Mesh**.

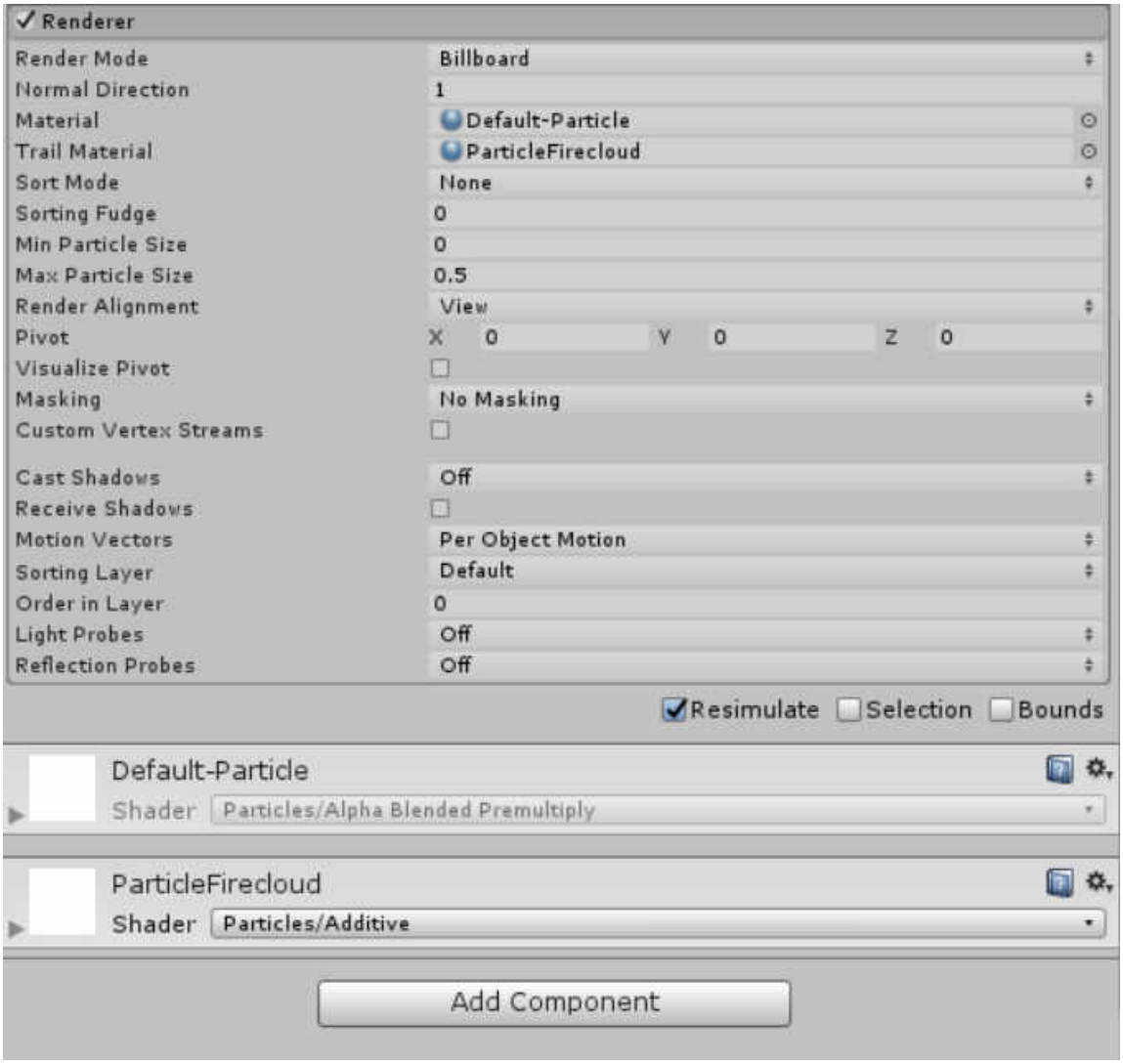

Рисунок 5.12 – Секція **Renderer**

Крім налаштування режиму рендеру, частки потребують в матеріалі, який буде призначений Мешу.

Можна використовувати попередньо підготовлені матеріали для систем частинок, що поставляються в складі активів **Unity**. Для цього необхідно перетягнути вибраний матеріал в чарунку матеріалу **Material** в інспекторі об'єктів.

Швидкість частинки в період її існування визначається в секції **Velocity over Lifetime**.

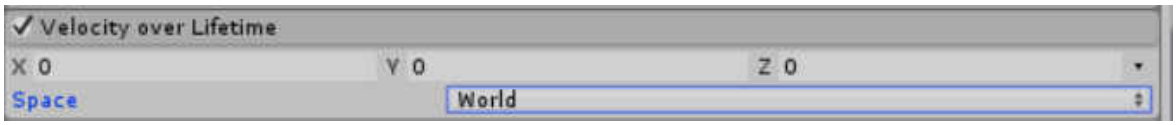

Рисунок 5.13 – Секція **Velocity over Lifetime**

Секцію зміни кольору протягом існування частинок **Color over Lifetime** можна використовувати для зміни кольору частинок в часі.

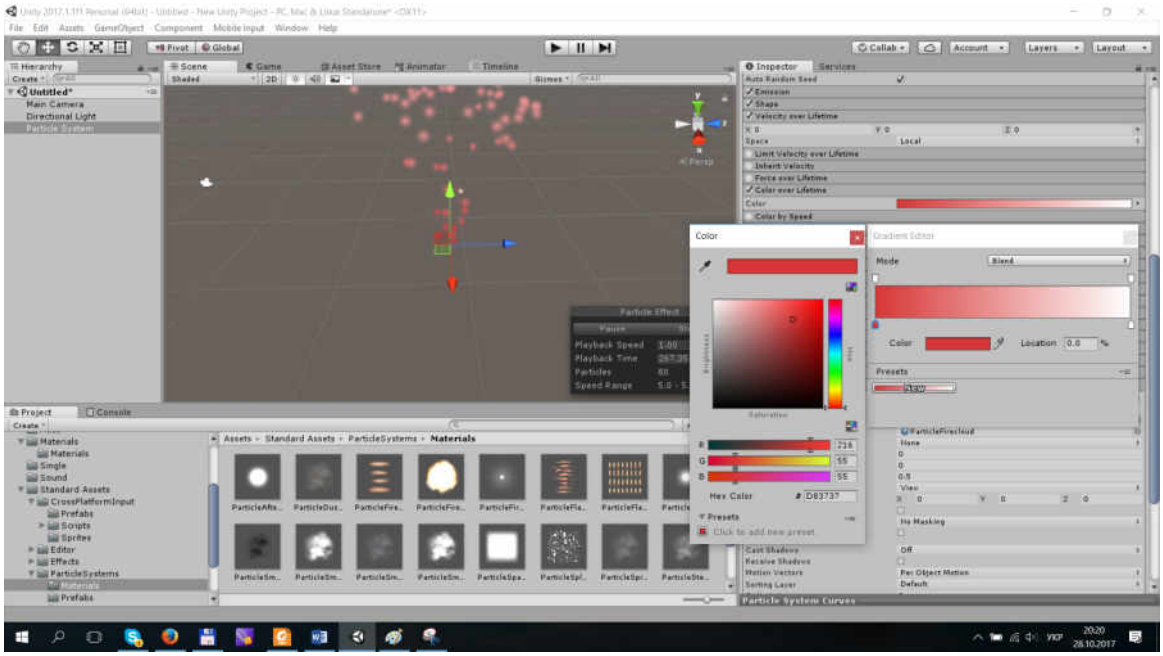

Рисунок 5.14 – Секція **Color over Lifetime**

Анімація за допомогою спрайтів

Перед початком створення анімації необхідно здійснити імпорт спрайтів. Після того як ви імпортували набір об'єктів спрайтів (не має значення, як окремі зображення або як текстуру атласу), ви готові запустити анімацію з їх участю.

На панелі проекту Unity виберіть всі спрайти, що входять в повну послідовність анімації, а потім перетягніть їх всі разом на панель ієрархії сцени. Якщо ви зробите це, з'явиться діалогове вікно збереження **Save**,

пропонуючи вам створити нову послідовність анімації (файл з розширенням **.anim**). Задайте ім'я файлу анімації (наприклад, PlayerRun.anim) і виберіть кнопку **Save**.

У сцені буде створено об'єкт спрайту, він повинен стати видимим на вкладках сцени і гри. Якщо ви протестуєте вашу сцену, то ваш персонаж буде анімаційним, ви повинні побачити всі зображення послідовності. Зазначимо, що Unity створило новий актив Animation Clip (.anim), який визначає ключову послідовність кадрів. Крім того, Unity створило контролер Mecanim для запуску відтворення анімації при завантаженні рівня і для управління швидкістю відтворення, додало компонент аніматора до об'єкта спрайту в сцені, щоб зв'язати об'єкт з його даними по анімації.

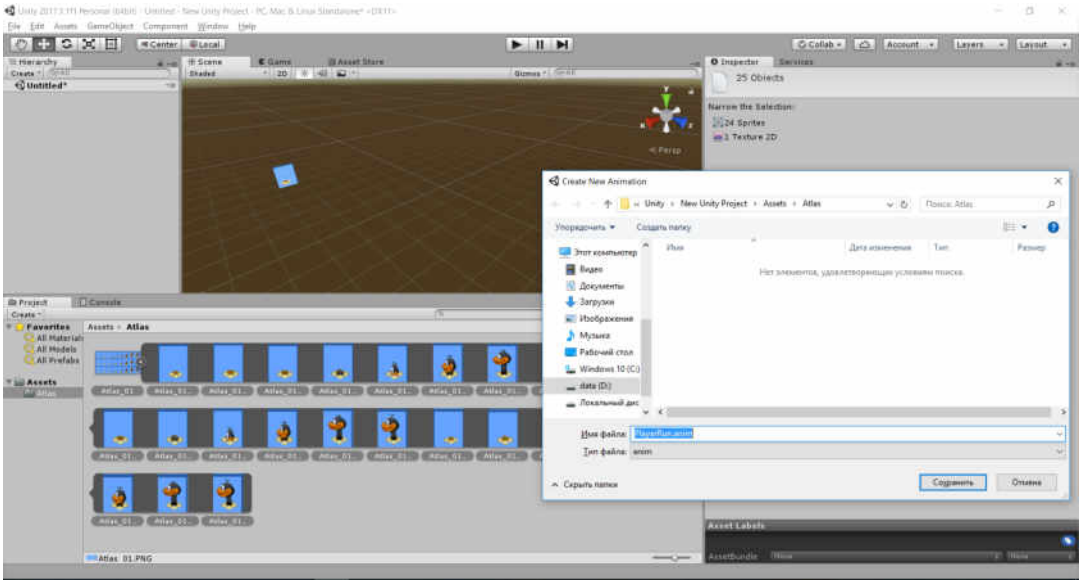

Рисунок 5.15 – Створення файлу анімації

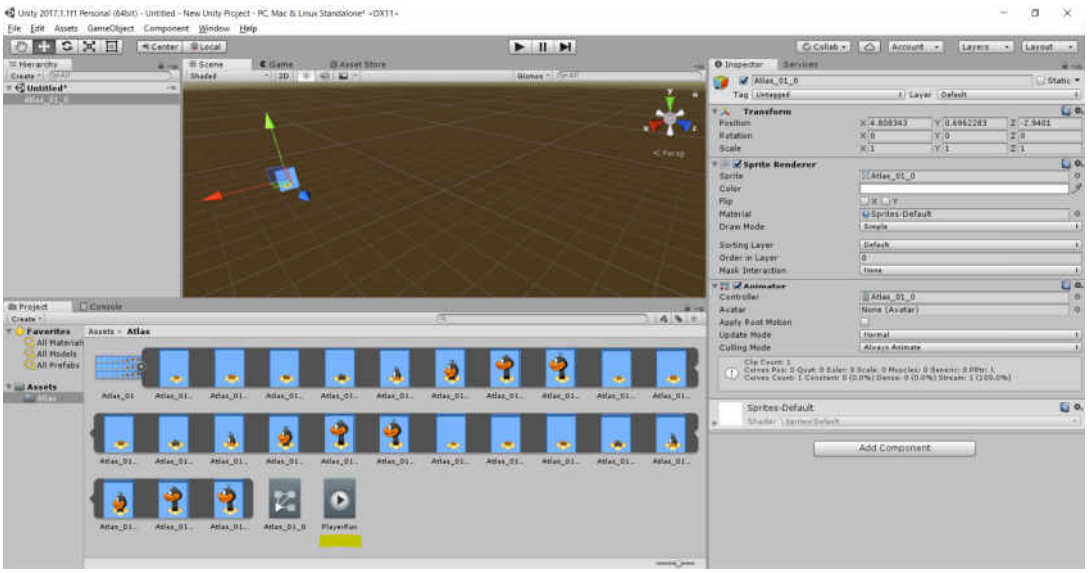

Рисунок 5.16– Створення нового активу Animation Clip (.anim) та контролеру Mecanim

Налаштування анімації за допомогою інструменту Mecanim

Якщо анімація спрайтами відтворюється занадто швидко або занадто повільно, то ви повинні будете відредагувати діаграму спрайту інструменту Mecanim.

Для досягнення цієї мети виберіть об'єкт спрайту в сцені. В інспекторі об'єктів двічі клацніть на актив **Animation Controller** всередині слота **Controller** компонента **Animator** .

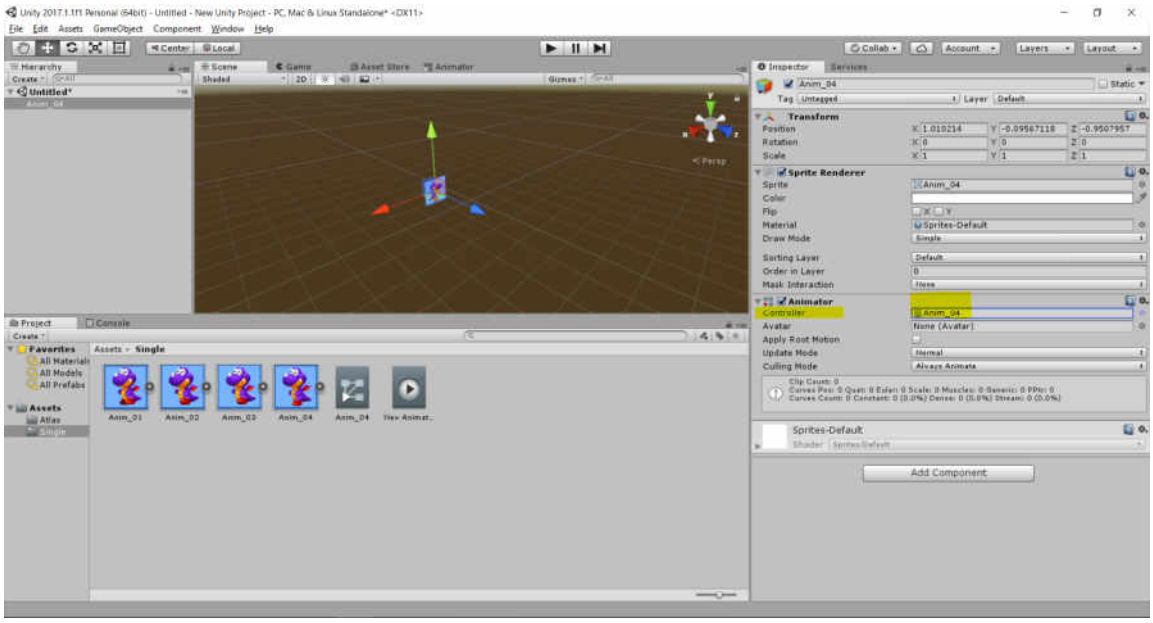

Рисунок 5.17– Налаштування анімації

Подвійний вибір активу **Animation Controller** спрайту викликає діаграму інструменту **Mecanim** для спрайту. З неї можна керувати швидкістю анімації. Ця діаграма містить кілька вузлів, пов'язаних між собою.

У діаграмі натисніть на вузол **Anim_04**, який представляє в діаграмі спрайт анімації, щоб вибрати його і переглянути його властивості в інспекторові об'єктів.

Швидкість анімації регулюється за допомогою параметра **Speed**. Значення 0 - зупинка, значення 1 - швидкість за замовчуванням, значення 0,5 - половинна швидкість, 2 - подвійна швидкість і т.д. Якщо анімація занадто повільна, то збільште значення швидкості, і якщо вона занадто швидка, то зменшите значення швидкості. Після того як ви закінчите, просто запустіть гру, щоб побачити ефект.

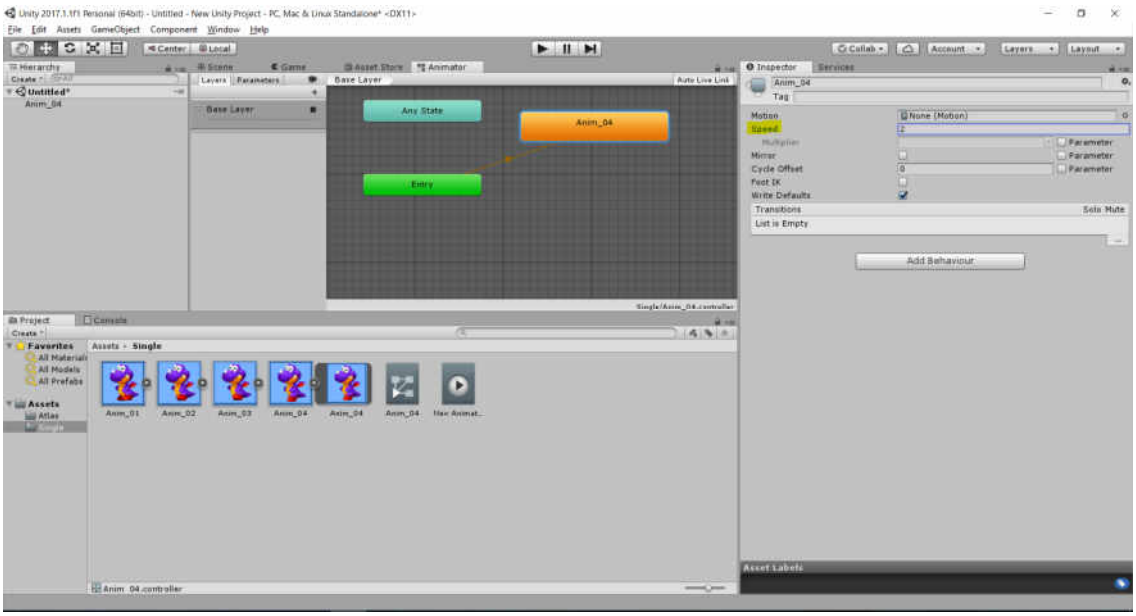

Рисунок 5.18– Діаграма інструменту **Mecanim**

За замовчуванням спрайт анімації зациклений, тобто відтворюється знову і знову без кінця. Коли відтворення анімації завершує цикл, воно просто повертається до початку, і відбувається нове відтворення.

Іноді потрібно відтворити анімацію тільки один раз, а потім зупинити. Для цього потрібно отримати доступ до даних анімації (вони знаходяться всередині активу .anim) і відкоригувати їх властивості. Щоб зробити це, виберіть актив спрайту анімації в панелі проекту. Активи анімації помічені іконкою **Play** і мають ім'я, яке ви присвоїли їм при їх створенні. Після вибору активу скиньте прапорець **Loop Time** в інспекторі об'єктів. Тепер анімацію буде відтворено тільки один раз.

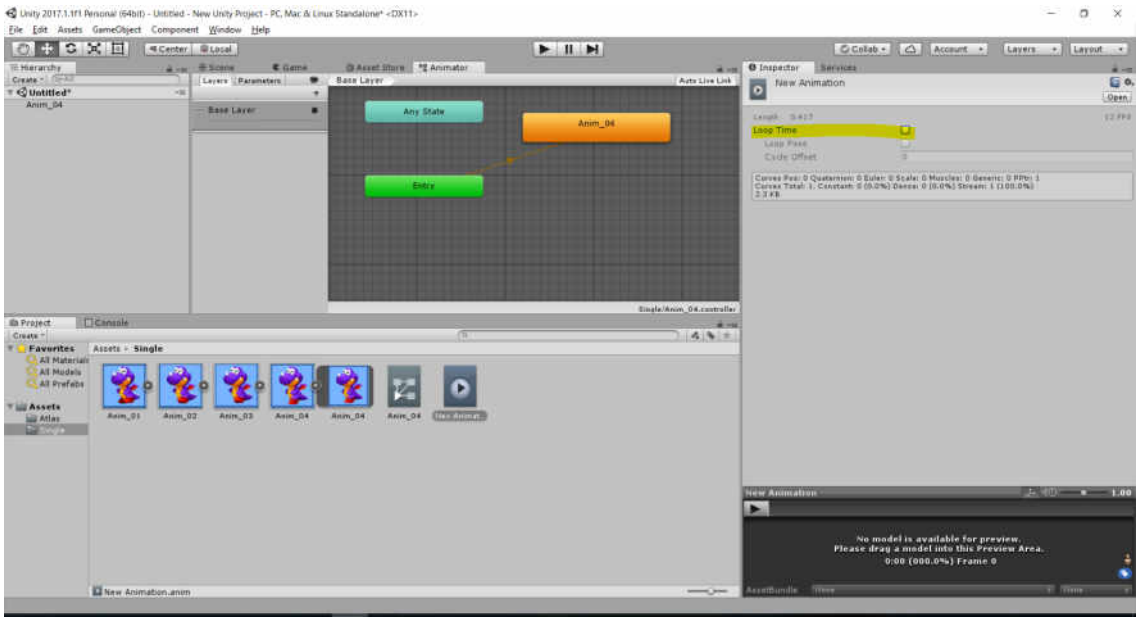

Рисунок 5.19– Налаштування анімації. Властивість **Loop Time**

Редагування кадрів анімації. Вікно Animation.

Якщо спрайт анімації має багато кадрів, то, можливо, при генерації анімації вони будуть розташовані в неправильному порядку, в результаті чого деякі кадри стануть з'являтися раніше або пізніше потрібного моменту.

Редагувати кадри анімації можна у вікні **Animation**. Це вікно можна отримати, перейшовши до **Window → Animation** в головному меню.

При відкритому вікні **Animation** виберіть об'єкт спрайту в сцені, і його анімаційні дані автоматично будуть відображені на часовій шкалі. Графік містить в собі весь період анімації, від початку і до кінця. Ромбовидні символи, рівномірно розподілені по шкалі, є ключові кадри, в яких змінюються зображення спрайтів. Ви можете вибрати конкретний ключовий кадр на часовій шкалі, обравши його. Потім ви можете переглянути його спрайтові властивості в інспекторові об'єктів.

.

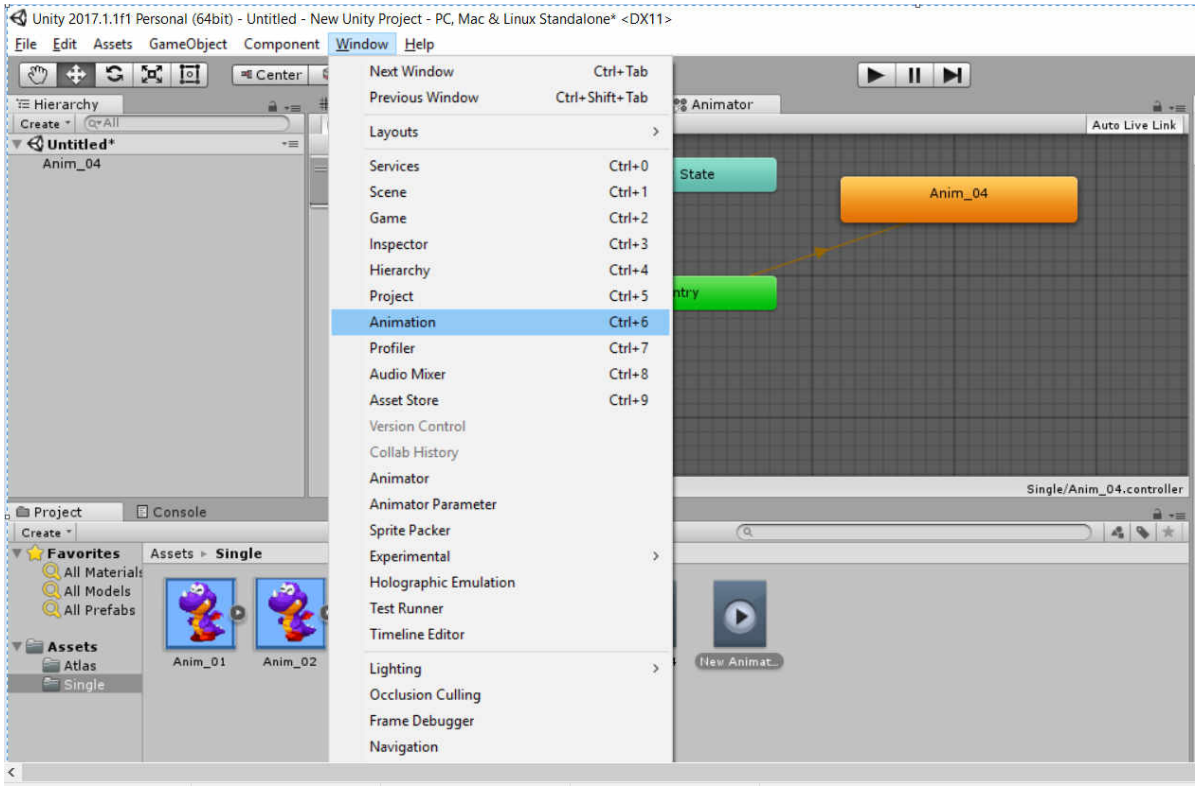

Рисунок 5.20– Вікно **Animation**

Вибравши ключовий кадр графіка, в якому знаходиться те зображення, яке не повинно бути показано в цей час, ви можете це легко виправити, натиснувши на поле **Sprite** в інспекторі об'єктів і вибравши новий спрайт в браузері спрайтів. Unity автоматично внесе потрібні зміни, і обраний вами спрайт буде відображатися в цьому ключовому кадрі.

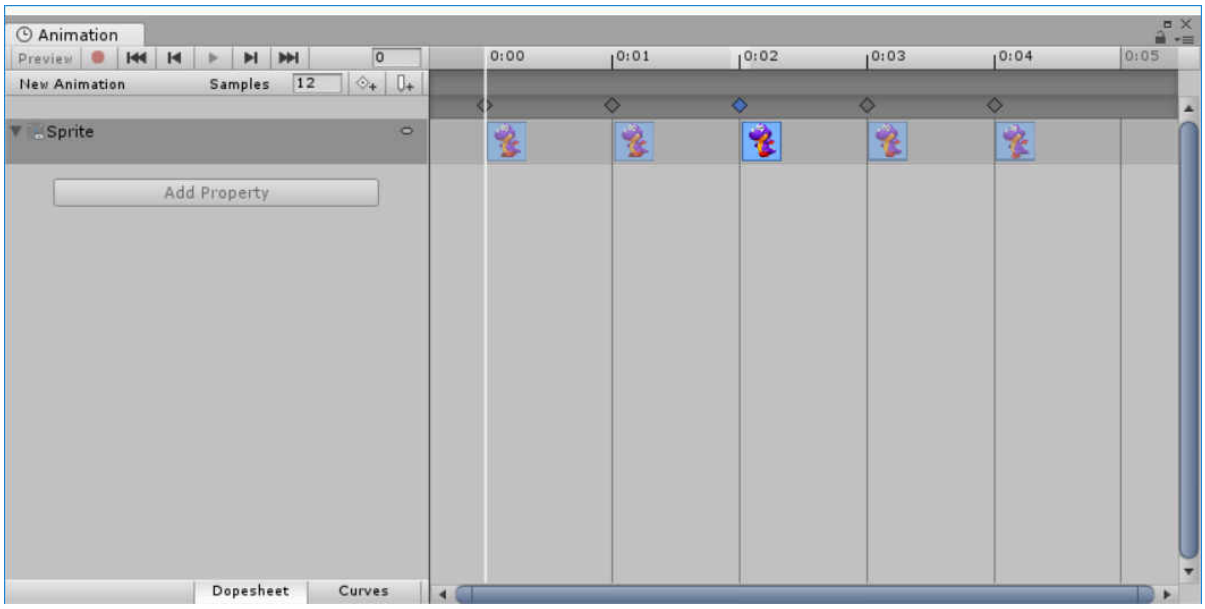

Рисунок 5.21– Ключові кадри анімації

Анімація персонажів

Імпортуємо заздалегідь підготовленого персонажа з «**Asset Store**» за допомогою меню «**Window → Asset Store**» (Ctrl + 9).

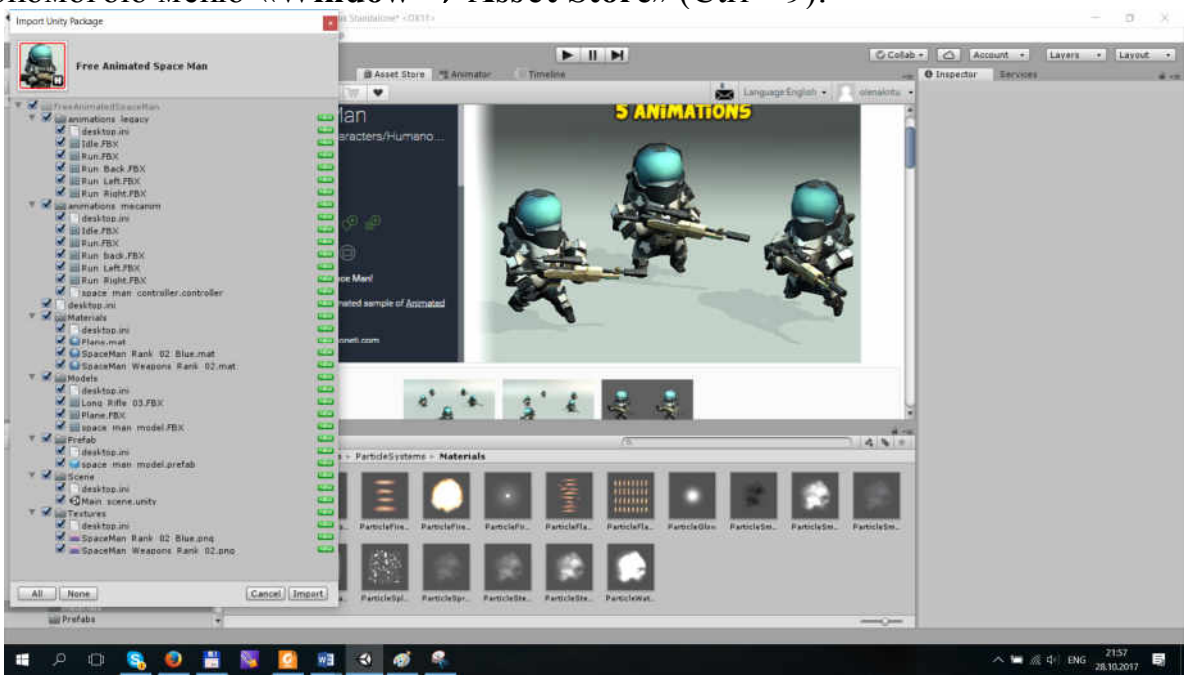

Рисунок 5.22– Імпорт персонажа з «Asset Store»

Після імпорту знаходимо у вікні «**Project**» префаб персонажа «FreeAnimatedSpaceMan». Перетягуємо цей префаб на сцену, і бачимо наш персонаж.

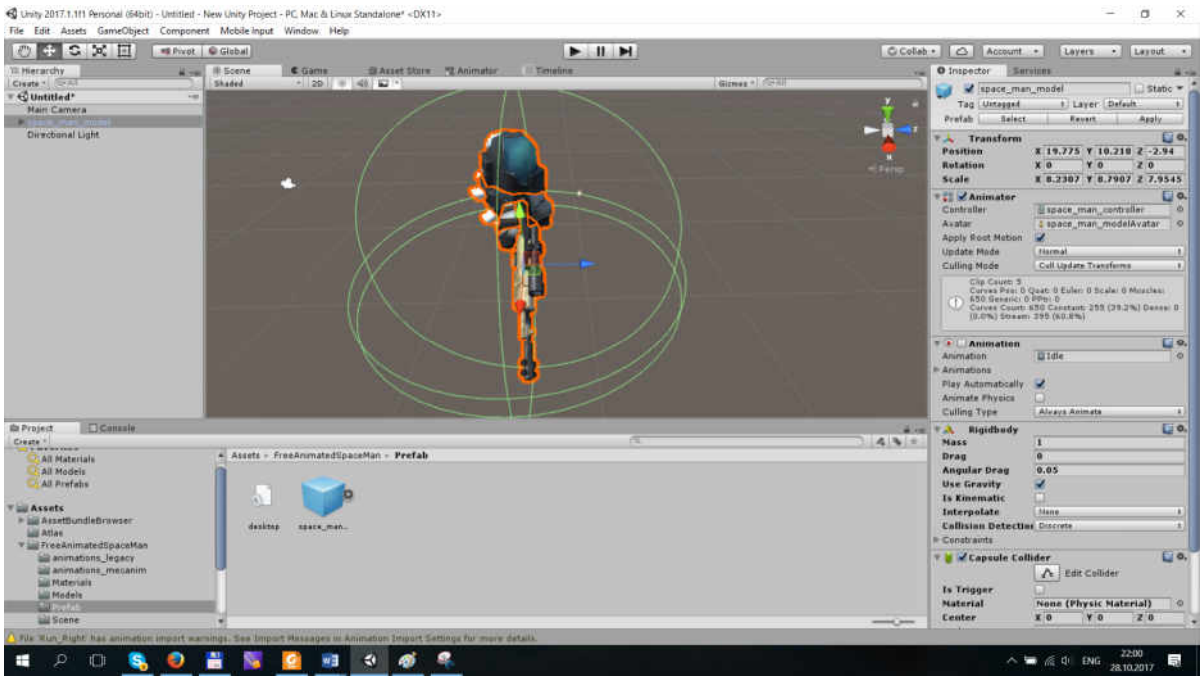

Рисунок 5.23– Персонаж «FreeAnimatedSpaceMan»

Натискаємо кнопку «**Add component → Physics → Rigidbody**» і «**Add component → Physics → Capsule Collider**». У «**Capsule Collider**» виставляємо розміри капсули у відповідності з розмірами персонажа.

У компонента «**Rigidbody**» активуємо властивість «**Use gravity**». Розкриваємо вкладку «**Constraints**» і вибираємо всі 3 галочки на «**Freeze Rotation**». Це потрібно для того що б персонаж не провалювався крізь землю.

Далі необхідно створити скрипт переміщення персонажа.

```
using UnityEngine;
using System.Collections;
public class Move: MonoBehaviour {
public GameObject player;
public int SRotation = 5;
public int sp = 7;
public AnimationClip SpaceMan;
public int JSpeed = 30;
void Start () {
 player = (GameObject)this.gameObject;
 animation.AddClip(SpaceMan, " FreeAnimatedSpaceMan");
 }
 void Update(){
if (Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.UpArrow))
 {
        player.transform.position += player.transform.forward * sp *
Time.deltaTime;
        animation.CrossFade("FreeAnimatedSpaceMan");
 }
```

```
        if (Input.GetKey(KeyCode.S) || Input.GetKey(KeyCode.DownArrow))
         {
              player.transform.position -= player.transform.forward * sp *
Time.deltaTime;
         }
        if (Input.GetKey(KeyCode.A) || Input.GetKey(KeyCode.LeftArrow))
         {
              player.transform.Rotate(Vector3.down * SRotation);
         }
         if (Input.GetKey(KeyCode.D) || Input.GetKey(KeyCode.RightArrow))
         {
              player.transform.Rotate(Vector3.up * SRotation);
         }
        if (Input.GetKeyDown(KeyCode.Space))
         {
              player.transform.position += player.transform.up * JSpeed *
Time.deltaTime;
         }
        }
        }
```

Після створення скрипта його необхідно додати до 3D-об'єкту шляхом перетягування скрипта на об'єкт в сцені.

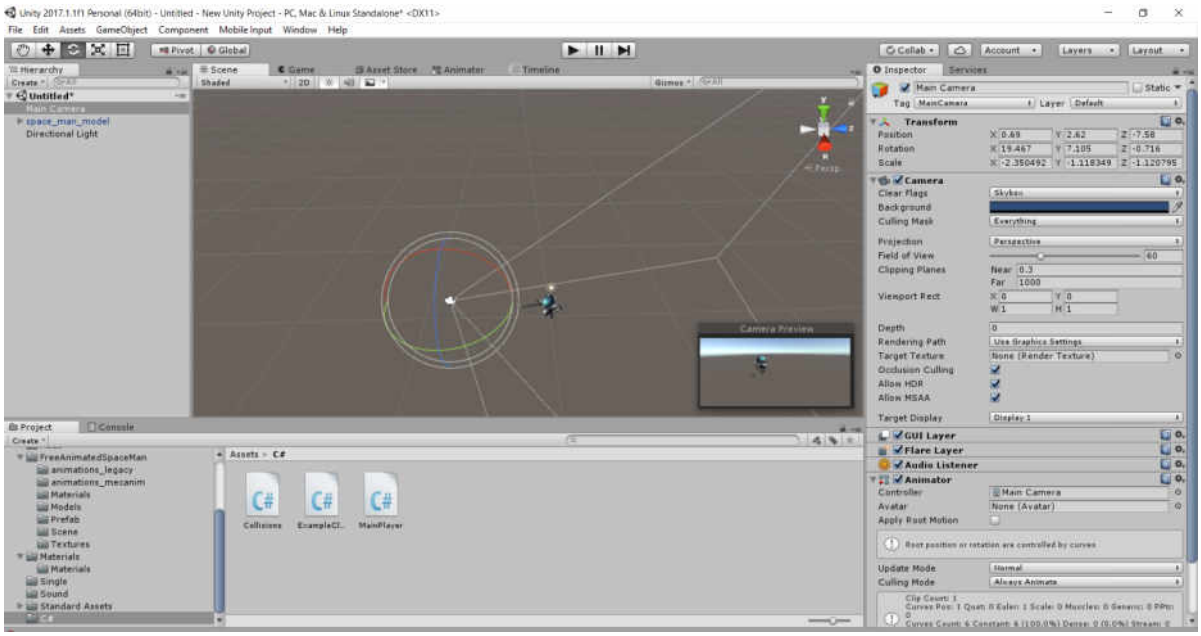

Рисунок 5.24– Персонаж «FreeAnimatedSpaceMan»

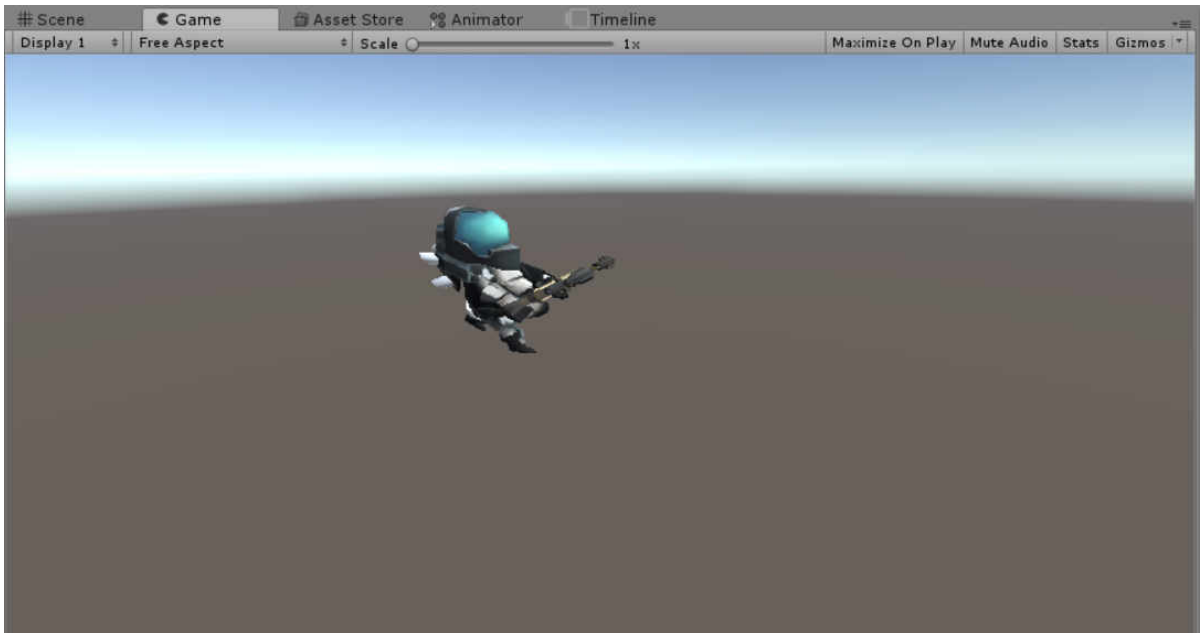

Рисунок 5.25– Анімація персонажу «FreeAnimatedSpaceMan»

Анімована текстура (Movie Texture)

Movie Textures це анімовані **Textures**, створені з відеофайлів. Переміщаючи відео файл в папку Assets Folder вашого проекту, ви можете використовувати імпортоване відео як звичайну текстуру.

Відео імпортуються через **Apple QuickTime**. Формати файлів, які може відтворювати **QuickTime** (зазвичай .mov, .mpg, .mpeg, .mp4, .avi, .asf). При імпортуванні відео в Windows, потрібно мати встановлений в системі **Quicktime** програвач.

При додаванні в проект відеофайлу, він буде автоматично імпортований і перетворений в формат **Ogg Theora format**. Відразу, після імпорту анімованої текстури, ви можете призначити її будь-який **GameObject** або **Material**, точно також як і будь-якій звичайній текстурі.

Програвання ролика

Анімована текстура не буде автоматично програватися при запуску гри. Для цього потрібно створити скрипт.

```
((MovieTexture)GetComponent<Renderer>().material.mainTexture).Play();
```

Далі необхідно призначити необхідний скрипт параметру «Включити відтворення текстури» (**toggle Movie playback**) після натискання пробілу:

```
public class PlayMovieOnSpace : MonoBehaviour {
   void Update () {
```

```
        if (Input.GetButtonDown ("Jump")) {
            Renderer r = GetComponent<Renderer>();
            MovieTexture movie = (MovieTexture)r.material.mainTexture;
            if (movie.isPlaying) {
                movie.Pause();
            }
            else {
                movie.Play();
            }
        }
    }
}
```

Контрольні запитання

- Для чого в Unity 3D використовуються система Mecanim?
- Яким чином в Unity 3D здійснюється анімація спрайтів?
- Назвіть типи анімації об'єктів в Unity 3D.
- Що таке атлас спрайтів?
- Яким чином в Unity 3D здійснюється відтворення відеофайлів у вигляді анімованих текстур?

ЛЕКЦІЯ №6
ТЕМА: «ТЕХНОЛОГІЯ РОЗРОБКИ КРОС-ПЛАТФОРМНИХ КОМП’ЮТЕРНИХ ІГОР В UNITY 3D»

Анотація

Лекція знайомить з технологією створення ігрових додатків для різних платформ (Windows, Android) в Unity 3D.

Мета лекції

Ознайомити студентів з технологією створення крос-платформних комп’ютерних 3D-ігор (Windows, Android).

Очікувані результати

Сформувати у студентів знання щодо технології розгортання ігрових додатків на різних платформах без прив'язки до Unity 3D.

Поняття крос-платформної комп’ютерної гри

Створені за допомогою Unity програми працюють під операційними системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а також на ігрових приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One і MotionParallax3D дисплеях (пристрої для відтворення віртуальних голограм), наприклад, Nettlebox.

Unity також має можливість створювати додатки для запуску в браузерах за допомогою спеціального модуля Unity (Unity Web Player) за допомогою реалізації технології WebGL.

Додатки, створені за допомогою Unity, підтримують DirectX і OpenGL.

Таким чином, за допомогою Unity можна створювати крос-платформні комп’ютерні ігри.

Під крос-платформністю мається на увазі здатність програмного забезпечення працювати більш ніж на одній апаратній платформі і (або) операційній системі.

Створення ігрового проекту для ОС Windows

Під створенням крос-платформних додатків мається на увазі генерація прикладних пакетів, які запускаються на різних платформах. На кожній платформі (Windows, Android та ін.) своя форма пакета, але як тільки ви згенерували виконуваний файл, з'являється можливість поширювати гру і грати в неї без прив'язки до Unity.

Один проект Unity можна розгорнути на різних платформах - його не потрібно кожного разу генерувати заново.

Подивитися всі варіанти платформ можна у вікні «**Build Settings**». Відкрити вікно «**Build Settings**» можна за допомогою меню «**File→Build Settings**» (рис. 6.1).

Вікно «**Build Settings**» містить мінімальні налаштування, необхідні для збірки проекту. Перше, що необхідно зробити - додати в проект ігрову сцену. Зробити це можна за допомогою кнопки «**Add Open Scenes**», або просто перемістити файл сцени з вікна проекту в поле «**Scenes In Build**».

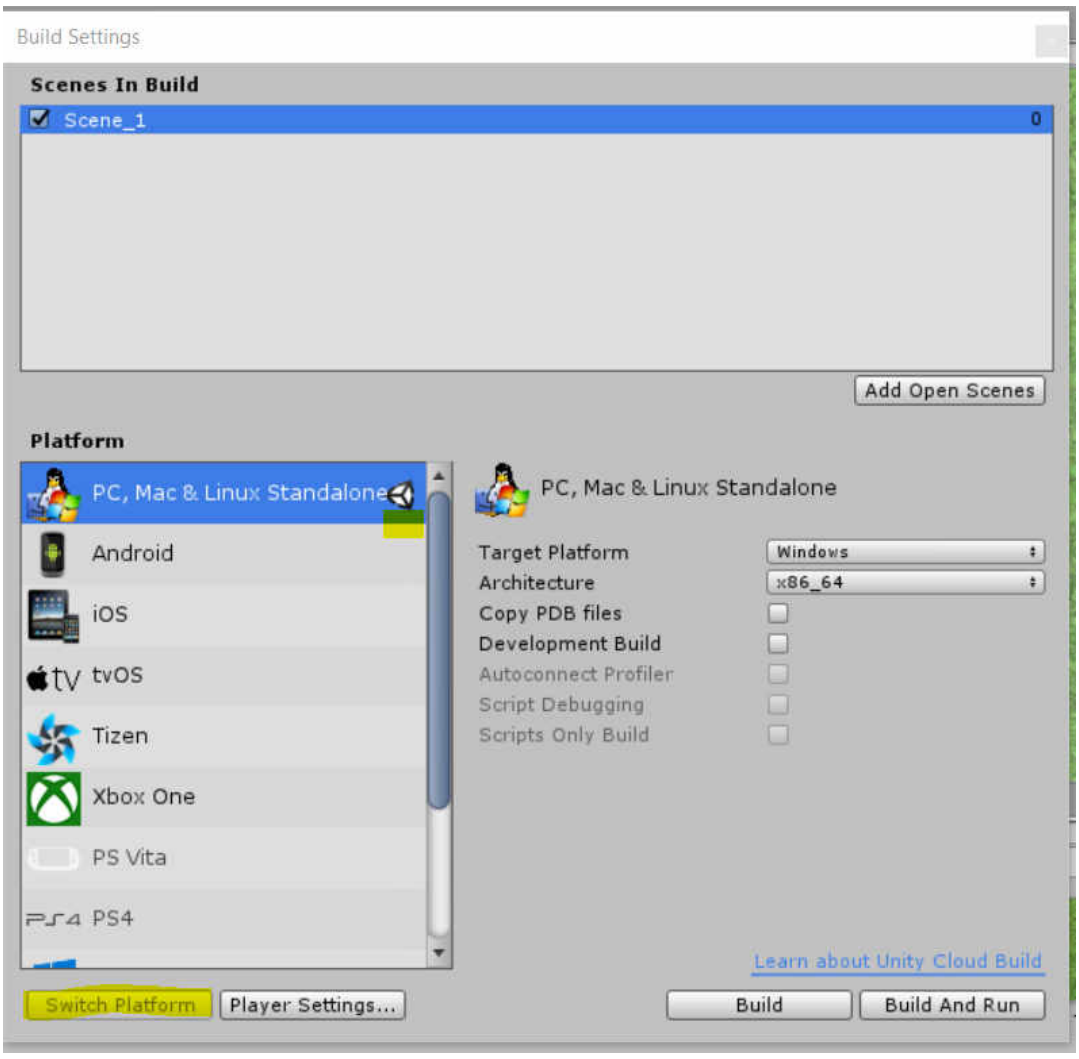

Рисунок 6.1– Вікно «**Build Settings**»

Зверніть увагу на цифру нуль, праворуч від назви файлу сцени. Це її порядковий номер. Якщо сцен в вашому проекті кілька - пам'ятаєте, що сцена, яка повинна буде завантажуватися першої, повинна отримати ідентифікатор нуль. Інакше, Unity автоматично завантажить замість вступного ролика сцену, що випадково отримала головний ідентифікатор.

Перелік платформ, які підтримує **Unity,** подано у лівій частині вікна. Активні платформи позначено значком **Unity.** Достатньо виділити платформу з цього переліку та натиснути кнопку «**Switch Platform**».

Далі праворуч на вкладці «**Target Platform**» вибираємо платформу «**Windows**», на вкладці «**Architecture**» вибираємо архітектуру, яка буде підтримуватися проектом.

Для генерації ігрового проекту у нижній частині вікна знаходяться кнопки «**Build**» і «**Build and Run**». Кнопка «**Build and Run**» відрізняється від кнопки «**Build**» тим, що автоматично запускає згенерований ігровий додаток.

Відкрити перелік налаштувань ігрового додатку на панелі «**Inspector**» можна за допомогою кнопки «**Player Settings**» (рис. 6.2). Ці налаштування контролюють різні аспекти готового додатка.

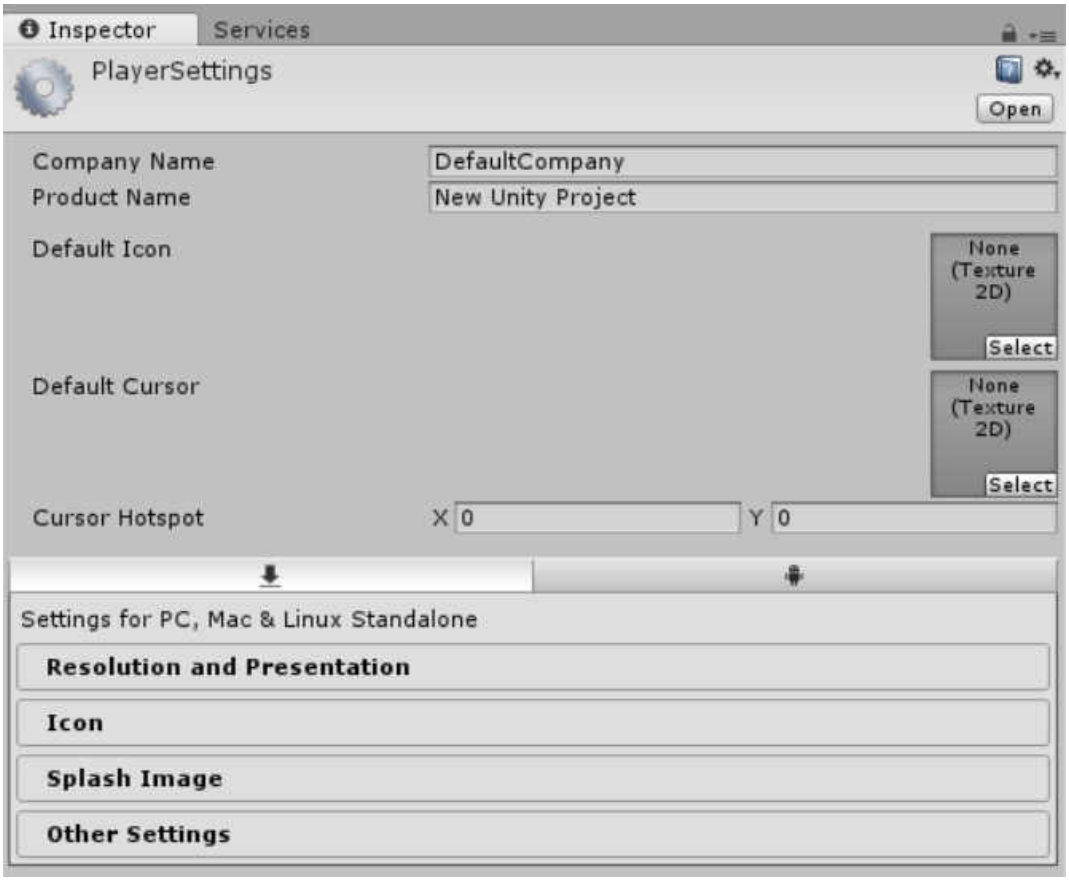

Рисунок 6.2– Вікно «**Player Settings**»

Поля «**Company Name**» і «**Product Name**» дозволять вказати назву гри і розробника. За допомогою «**Default Icon»** можна встановити іконку ігрового додатку. Для цього необхідно обрати зображення (при необхідності імпортувати його) на вкладці «**Project**» та перетягнути його у розділ «**Texture 2D**» на панелі «**Inspector**». Крім того, за допомогою «**Default Cursor»** можна також задати зображення курсора.

Після натискання на кнопку **«Build»** відкриється вікно вибору файла, в якому потрібно вказати адресу для генерації пакета ігрового додатку. Відразу після вказівки місця розташування почнеться процес побудови, після чого **Unity** створить виконуваний файл для активної в даний момент платформи. У нашому випадку це платформа «**Windows**».

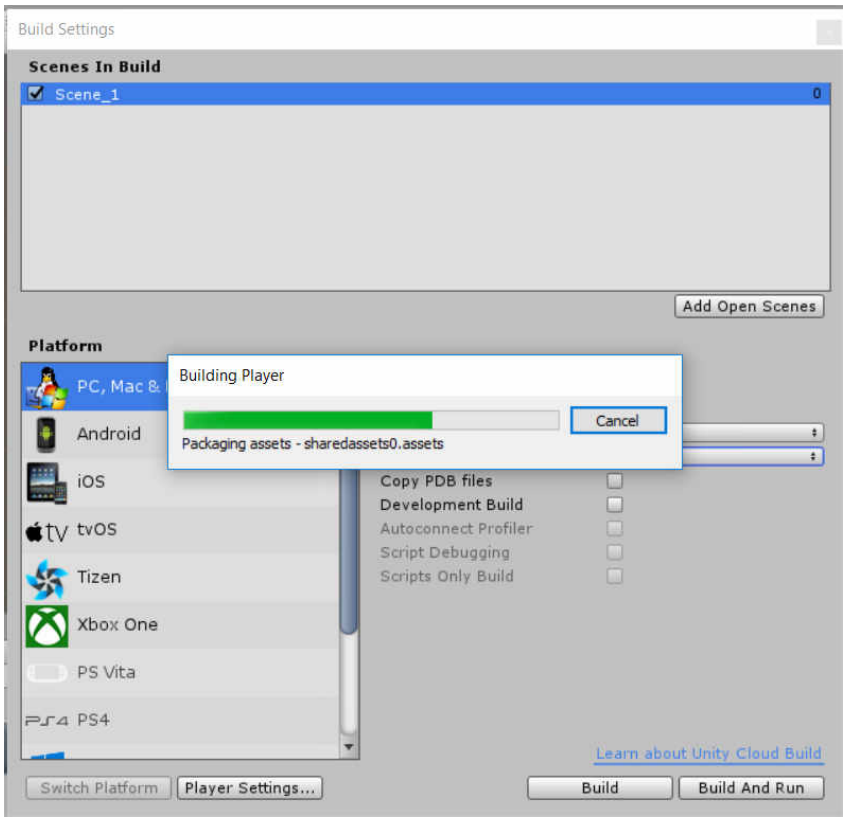

Рисунок 6.3– Процес побудови ігрового проекту

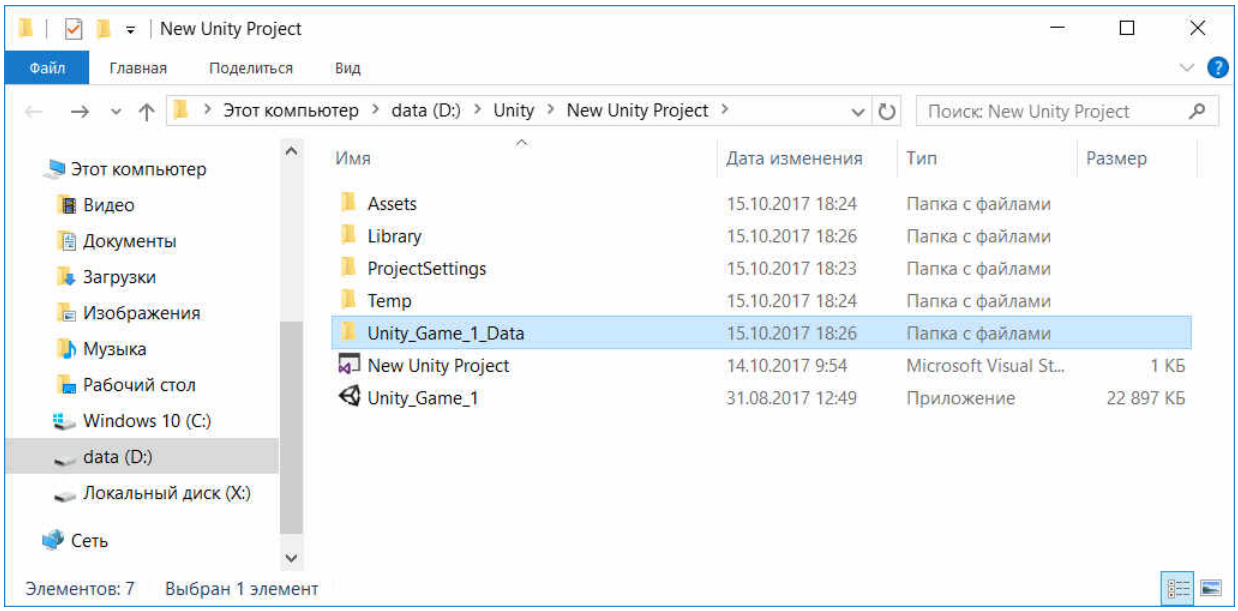

Рисунок 6.4– Файли проекту для платформи «**Windows**»

На додаток також впливають налаштування, доступ до яких здійснюється за допомогою меню «**Edit**». Зокрема, саме тут налаштовується візуальна якість готового додатка.

Виберіть в меню «**Edit**» команду «**Project Settings**», а потім в додатковому меню - команду «**Quality**» (рис.6.5).

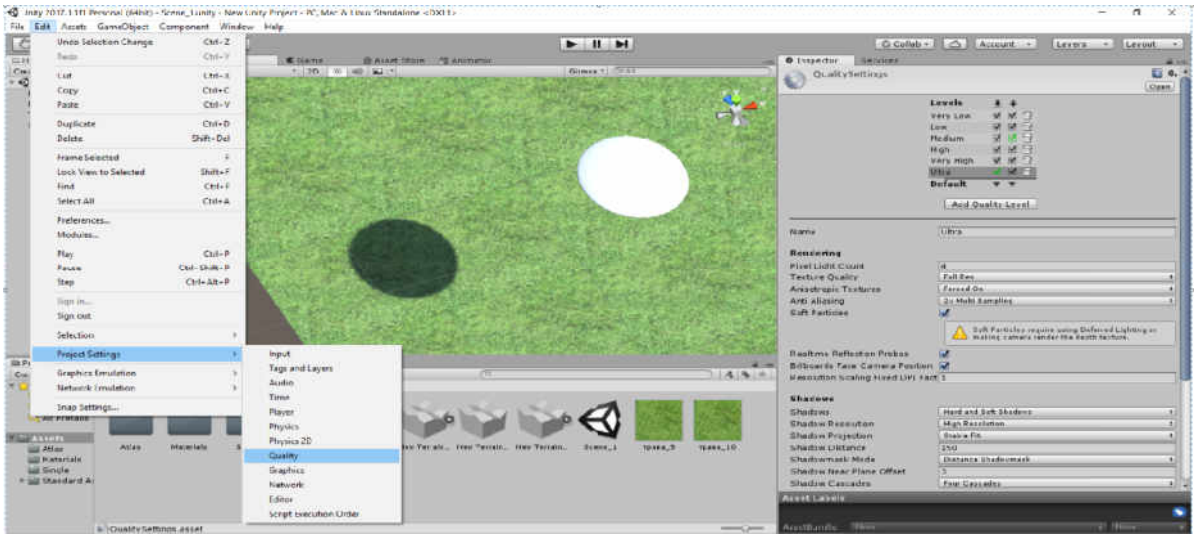

Рисунок 6.5– Меню «Edit»

На панелі «**Inspector**» з'являться елементи управління якістю, найбільш важливими з яких є прапорці в розташованій зверху групі. У верхньому рядку знаходяться значки можливих платформ, а збоку вказані варіанти налаштувань якості. Встановлені прапорці показують доступні для даної платформи налаштування, а ті, що виділені зеленим - поточні налаштування.

У більшості випадків за замовчуванням застосовується варіант «**Very Low**» (мінімальна якість), але можна вибрати варіант «**Very High**» або «**Ultra**» (максимальна якість) (рис. 6.6).

Рисунок 6.6– Елементи управління якістю

Створення ігрового проекту для ОС Android

Unity має можливість генерувати файли формату **APK** (Android Application Package). Для цього необхідно додати в **Unity** шлях до середовища розробки **Android SDK**, яке вже містить необхідний компілятор.

Android - операційна система (ОС) для смартфонів, планшетних комп'ютерів, електронних книг, цифрових програвачів, нетбуків, смартбуків, очок Google, телевізорів, систем автоматичного керування автомобілем та іншими пристроями. ОС базується на Linux-ядрі та власної реалізації віртуальної машини Java від Google.

На даний момент останньою версією є Android 7.1 Nougat, яка вийшла в жовтні 2016 року:

Таблиця 6.1 – Версії Android

Версія	Кодове ім'я	Дата випуску	Рівень API	Частка ринку (2017 рік)
7.0	Nougat	4 жовтня 2016	25	0,4%
7.0	Nougat	22 серпня 2016	24	2,4%
6.0	Marshmallow	5 жовтня 2015	23	31,3%
5.1	Lollipop	9 березня 2015	22	23,1%

5.0	Lollipop	3 листопада 2014	21	9,4%
4.4	KitKat	31 жовтня 2013	19	20,8%
4.3	Jelly Bean	24 липня 2013	18	1,5%
4.2	Jelly Bean	13 жовтня 2012	17	5,4%
4.1	Jelly Bean	9 липня 2012	16	3,7%
4.0	Ice Cream Sandwich	16 грудня 2011	15	1%
2.3	Gingerbread	9 лютого 2012	10	1%

Налаштування Android SDK

Android SDK – це середовище розробки додатків для операційної системи **Android**. **Android SDK** дозволяє створювати і тестувати **Android**-додатки, що використовують камеру мобільного пристрою, акселерометр, компас, дані GPS, доступ по Bluetooth, Wi-Fi, EDGE і 3G.

Android SDK підтримує роботу з мультимедійним контентом (аудіо, відео, зображення в форматах MPEG4, H.264, MP3, AAC, AMR, JPG, PNG і GIF), базами даних SQLite, інтегрованим браузером на движку WebKit, віртуальною машиною Dalvik, GSM телефонією і т.д. Користувачі **Android SDK** мають можливість тестувати створені ними додатки за допомогою вбудованого емулятора.

Завантажити **Android Studio з Android SDK** можна з офіційного сайту за посиланням **https://developer.android.com/studio/index.html**

Відкриємо **Android Studio**. Внизу стартового екрана програми знайдемо кнопку «Configure» і натиснемо на неї.

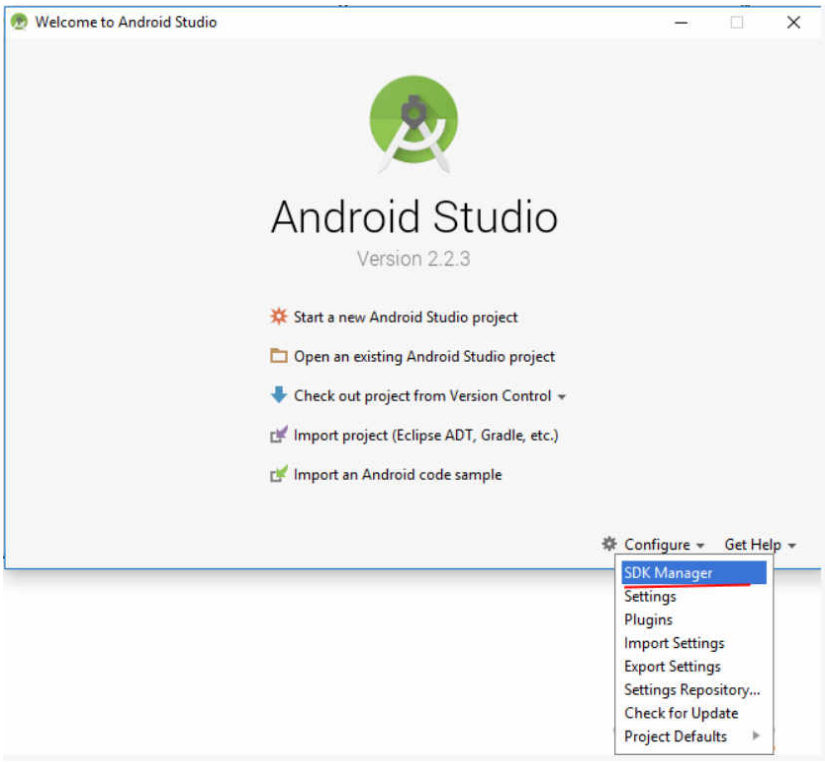

Рисунок 6.7 - Стартовий екран програми **Android Studio**

В меню натиснемо на пункт «SDK Manager». Після цього відкриється вікно з настройками для Android SDK Manager:

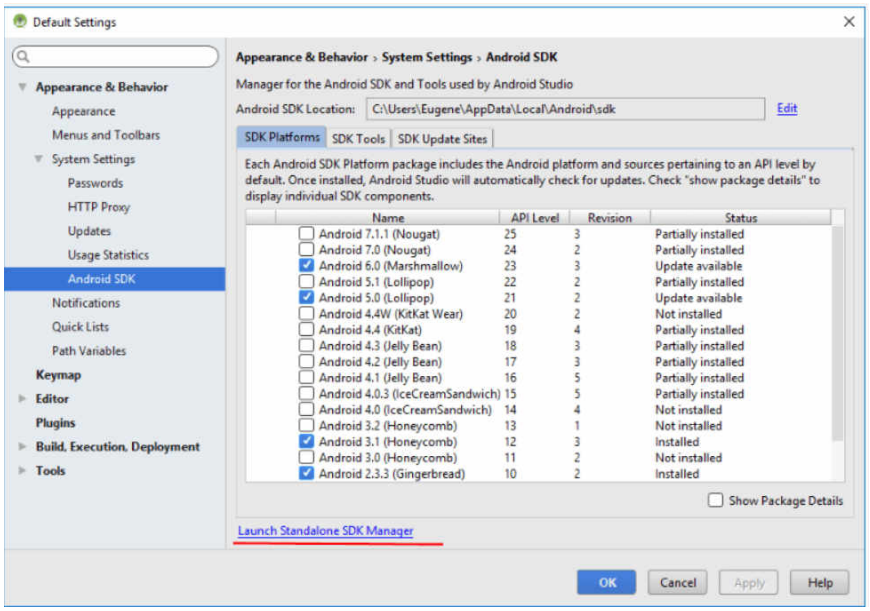

Рисунок 6.8 - Android SDK Manager

Тут ми можемо побачити, які версії **SDK** встановлені. Але кожна версія передбачає широке коло компонентів. Не всі ці інструменти можуть знадобитися, тому немає сенсу встановлювати всі версії SDK повністю.

Для більш детального перегляду всіх компонентів по кожній платформі натиснемо на посилання **Launch Standalone SDK Manager**, яке розташоване внизу вікна. Це посилання знову запустить **Android SDK Manager**, але в окремому вікні.

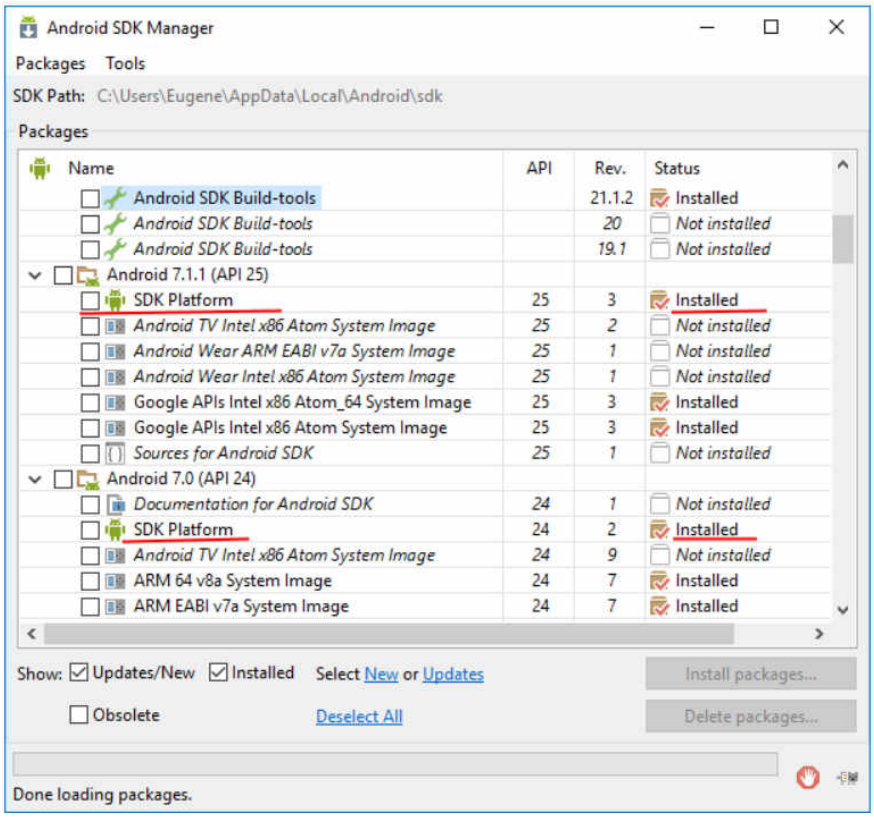

Рисунок 6.9 - Посилання Launch Standalone SDK Manager

Структурно **SDK Manager** побудований таким чином. Спочатку йде секція **Tools**. Вона містить пакети, необхідні для розробки програми.

Далі йдуть секції платформи **Android**. Тут ми детально можемо подивитися, які пакети для кожної платформи встановлені. В даному випадку нас перш за все буде цікавити пункт **SDK Platform**.

SDK Platform містить весь основний функціонал, який використовується при розробці. Даний пункт можна виділити для всіх тих платформ, під які ми збираємося компілювати Unity-додаток.

Крім **SDK Platform** кожна платформа, як правило, містить ще ряд компонентів:

Google API: різні API від Google, зокрема, інтерфейси по роботі з Google Maps, іншими сервісами Google.

ARM EABI v7a System Image: образ системи, що емулює роботу процесора ARM, необхідний для запуску емулятора.

Intel x86 Atom System Image / Intel x86 Atom_64 System Image: образ системи (для 86-х і 64-х платформ) від компанії Intel, також необхідний для запуску емулятора.

Google APIs ARM EABI v7a System Image: Google API для образу системи ARM EABI v7a System Image.

Google APIs Intel x86 Atom System Image / Intel x86 Atom_64 System Image: Google API для образу системи Intel x86 Atom System Image або Intel x86 Atom_64 System Image

Якщо тестування додатку буде відбуватися на емуляторі, то слід встановити для цього образ системи, який відповідає поточній робочій машині, і відповідне **Google API**.

Після всіх платформ в **SDK Manager** розташована секція **Extras**. Тут перераховані додаткові пакети, не прив'язані до платформ.

В секції **Extras** важливі такі пакети як:

Android Support Repository

Android Support Library

Google Play Services

Google Repository

Google Usb Driver

Ці пакети встановлюють репозиторії **Android** і сервіси **Google Play**. Крім того, якщо ви хочете використовувати для тестування додатків смартфон, то необхідно буде встановити USB-драйвер безпосередньо виробника смартфону.

Як правило, при підключенні смартфона система сама намагається встановити драйвер.

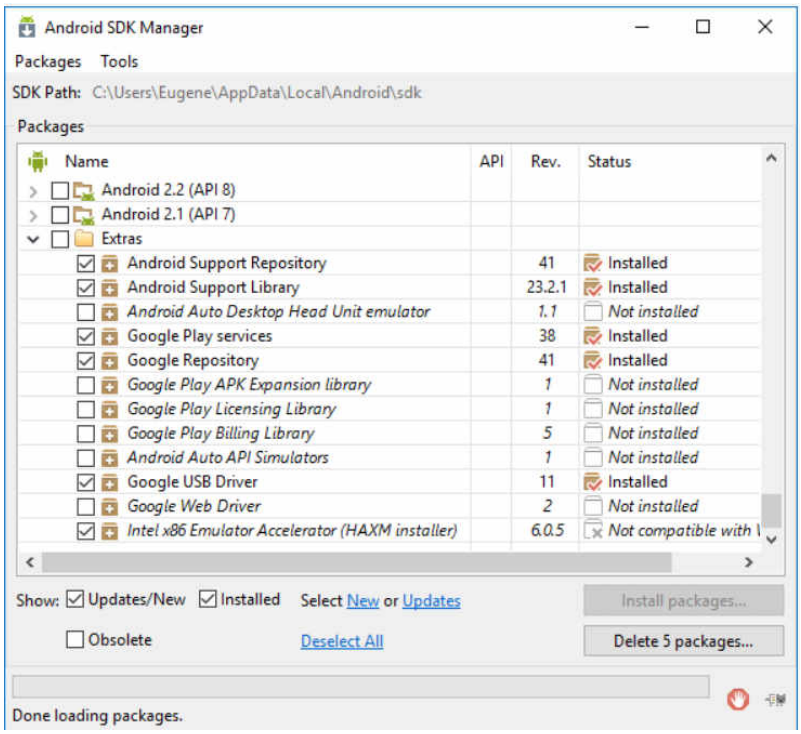

Рисунок 6.10 - Секція **Extras**

Вкажіть шлях до файлу **Android SDK** на вкладці «**External Tools**» у вікні «**Unity Preferences».**

Відкрити вікно «**Unity Preferences**» можна за допомогою меню **«Edit→ Preferences»**.

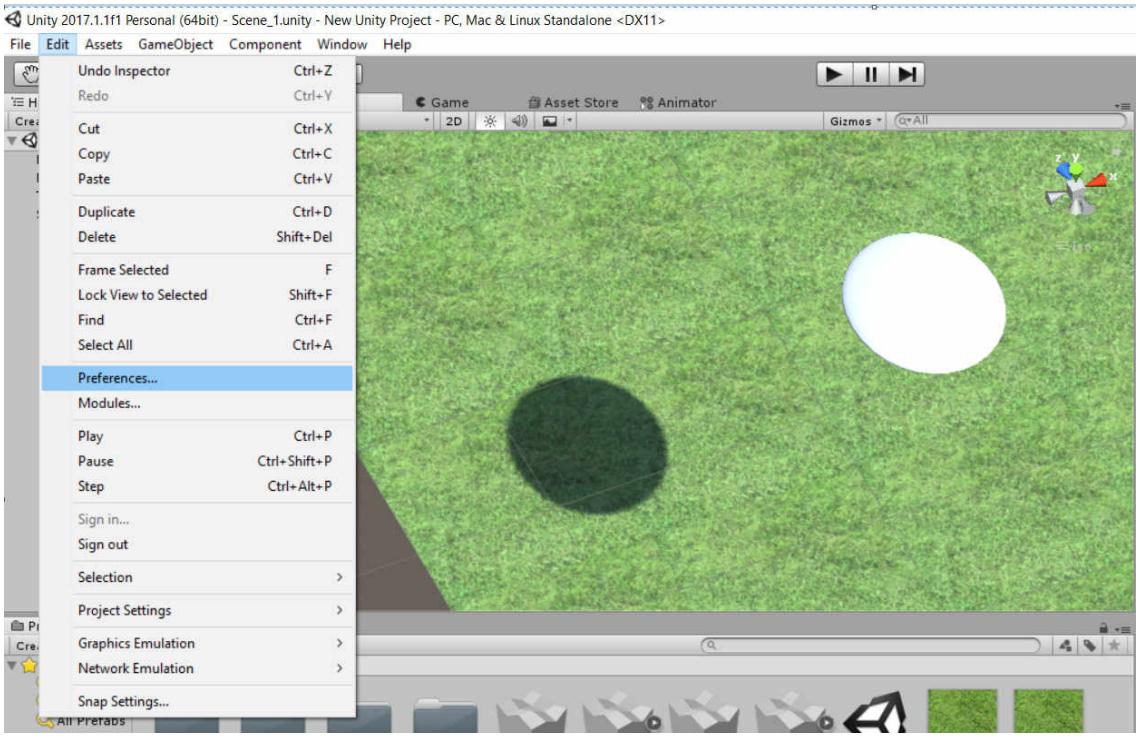

Рисунок 6.11– Меню **«Edit→ Preferences»**

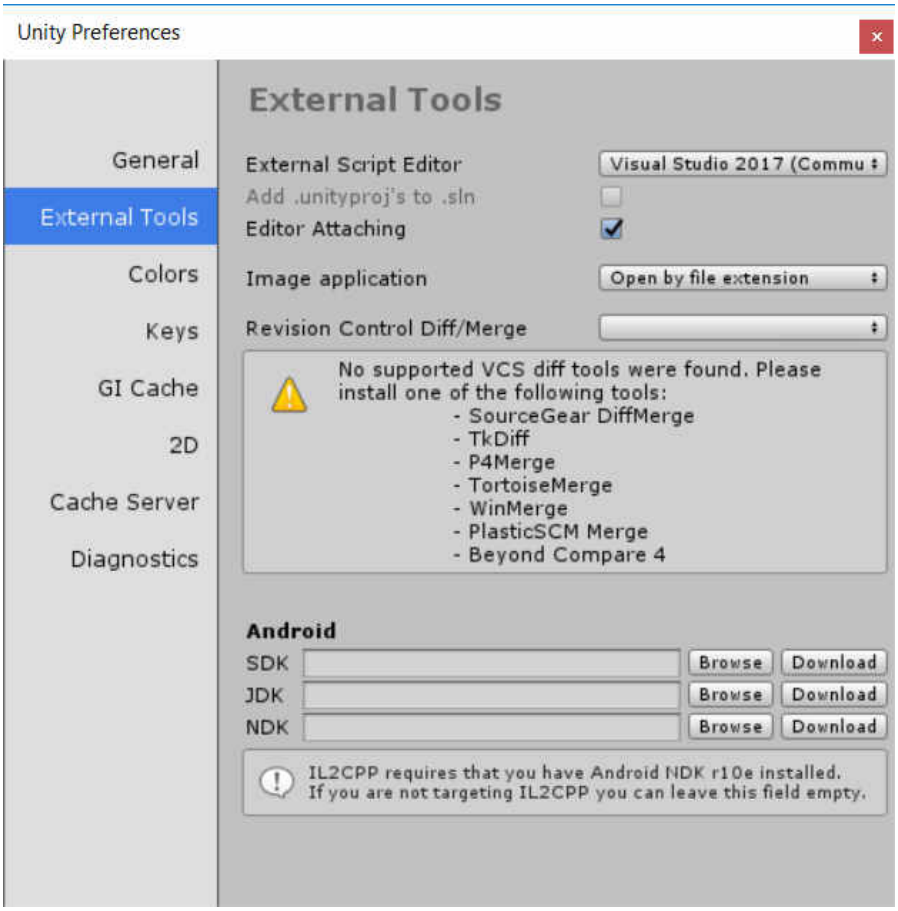

Рисунок 6.12– Вкладка «**External Tools**» у вікні «**Unity Preferences**»

Після цього необхідно задати параметр «**Package Name**» в розділі «**Other Settings**» на панелі «**Inspector**» у вигляді **com.назвакомпанії.назвапродукта** та запустить процес збірки ігрового проекту за допомогою кнопки «**Build**».

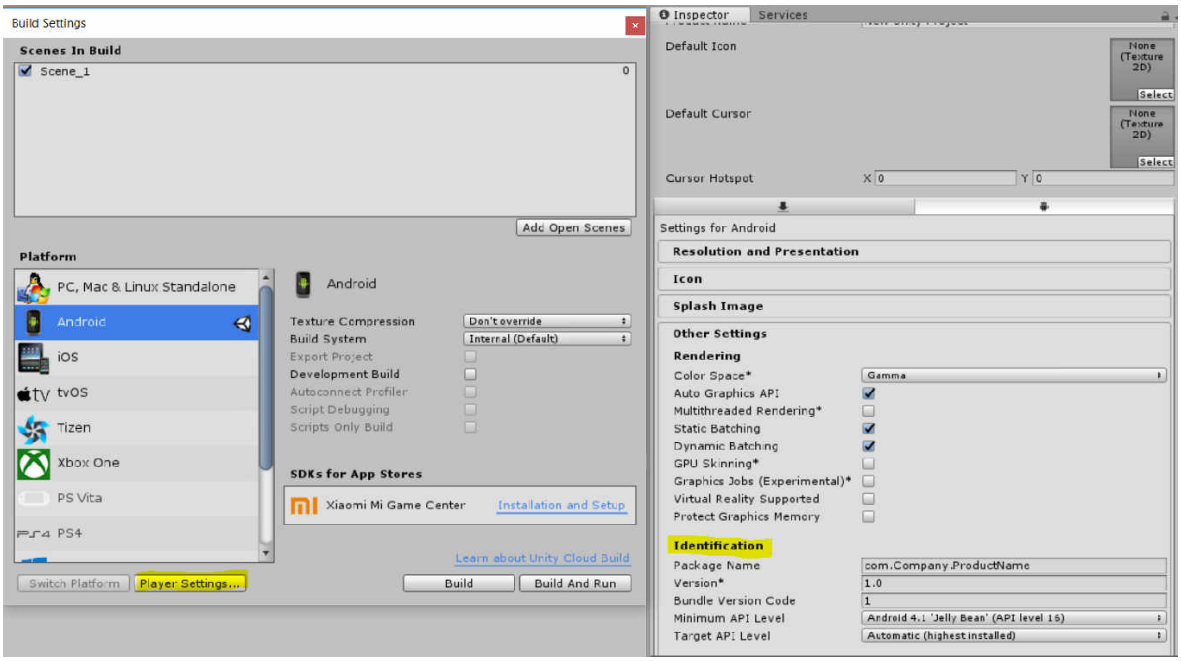

Рисунок 6.13– Налаштування параметру **«Package Name»**

Отриманий **APK** пакет ігрового додатку необхідно встановити на мобільний пристрій.

Контрольні запитання

- Для яких платформ можна створювати ігрові додатки в Unity 3D?
- Для чого використовується команда BuildSettings?
- Назвіть відмінності генерації пакетів для мобільних пристроїв.
- Для чого використовується параметр BundleIdentifer?
- Назвіть основні етапи налаштування інструментів збірки для Android?