Практична робота №2

Спрайтова 2D-анімація в Unity

Мета: ознайомитися з методами створення анімації в Unity, освоїти базові компоненти анімації, навчитися створювати безскриптову спрайтову анімацію.

Література

Керівництво Unity. https://docs.unity3d.com/
Unity Manual: https://docs.unity3d.com/Manual/AnimationSection.html
Animation: https://docs.unity3d.com/ScriptReference/Animation.html
Introduction to Sprite Animations: https://learn.unity.com/tutorial/introduction-to-sprite-animations#5fa66921edbc2a0020bcaadf

Зміст роботи

Завдання 1. Створити анімацію ігрового об'єкта із спрайтового атласу.

Методичні рекомендації

Система анімації в Unity дозволяє створювати чудово анімованих персонажів. Вона підтримує блендінг, мікшування, додавання анімацій, синхронізацію циклу ходьби, анімаційні шари, контроль всіх аспектів програвання (час, швидкість, ваги блендінгу), скіннінг мешів з 1, 2 або 4 кістками на вершину, а також засновані на фізиці rag-dolls (ганчіркові ляльки) і процедурну анімацію.

Створення анімованого персонажа включає в себе дві речі - переміщення в просторі сцени і відповідна анімація.

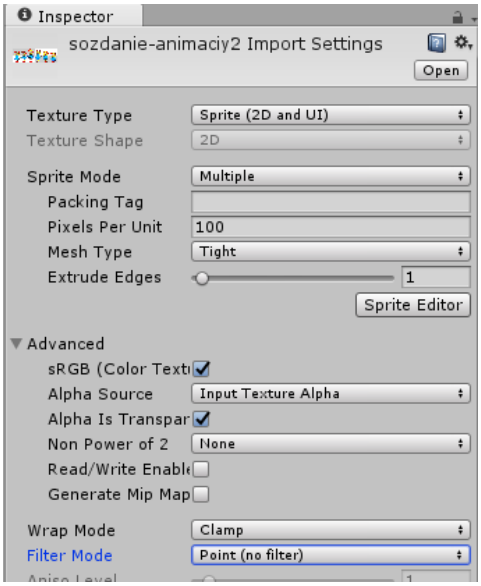
1. Створюємо новий 2d проект.
2. Вибираємо атлас зі спрайтами і перетягуємо його у **Project.**

3. Натискаємо на спрайт і виставляємо наступні значення в **Inspector:**

Texture Type: *Sprite (2d and UI)*

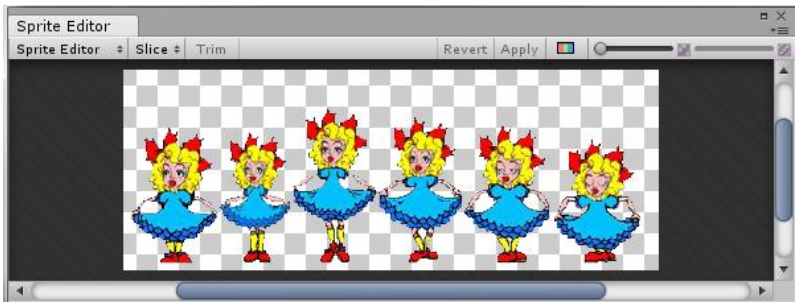
Sprite Mode: вид спрайту. Якщо спрайт має анімацію виставляємо-*Multiple*, якщо статичний спрайт то *Single*.

Filter Mode: за замовченням виставлено *Bilinear*, его не чіпаємо, якщо – піксельна графіка то модифікацію *Point*.



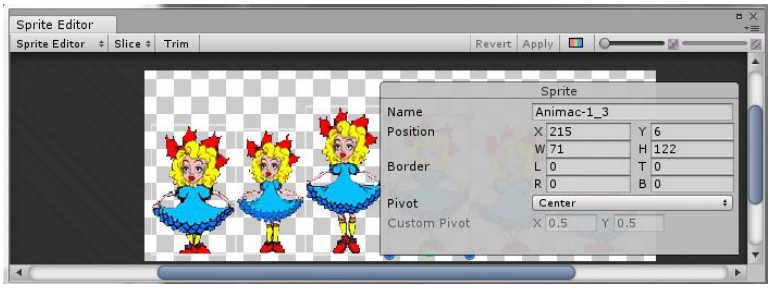
Натискаємо кнопку ***Apply***.

4. Натискаємо на *SpriteEditor* відкриється окреме вікно.

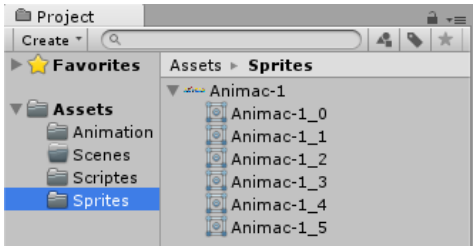


5. У вікні *SpriteEditor* натискаємо *Slice*, в *Type* ставимо *Automatic*.

Навіть якщо якийсь з кадрів вийшов трохи невдало - його можна відредагувати, клікнувши по ньому і змінивши значення висоти, ширини, розташування та інших параметрів у відповідному вікні або за допомогою мишки. Підтвердимо нарізку натисканням на кнопку *Apply* в правому верхньому кутку і закриваємо це вікно.



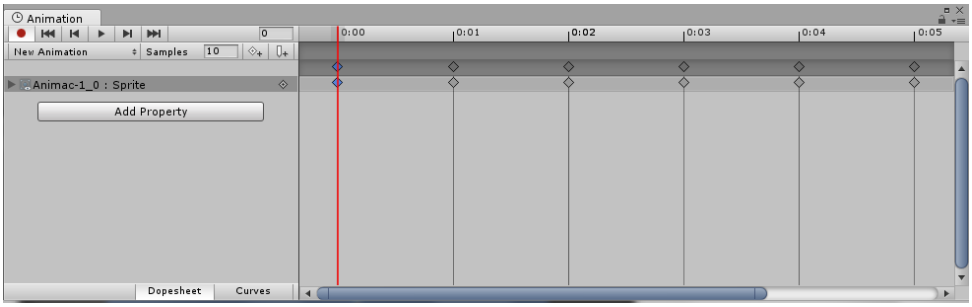
6. Якщо переглянути спрайт у Project, він має такий вигляд:



Створилися кадри для анімації.

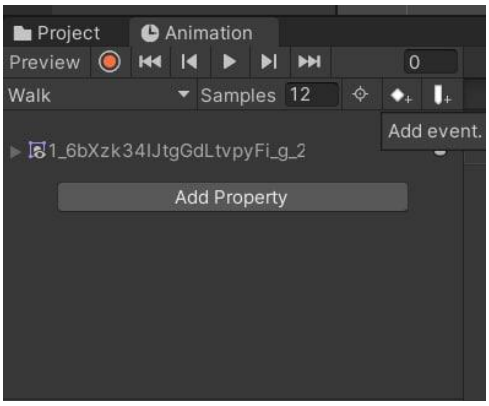
7. Перетягуємо спрайт на сцену.

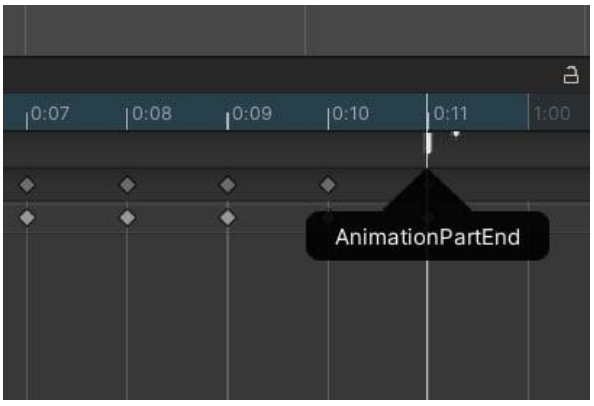
8. Натискаємо Ctrl+6 відкривається вікно *Animation* перетягуємо створені кадри. Зберігаємо анімацію. Натискаємо кнопку *Play* і дивимося, що отримали: якщо анімація програється дуже швидко тоді необхідно змінити значення *Sample* в *Animation*.

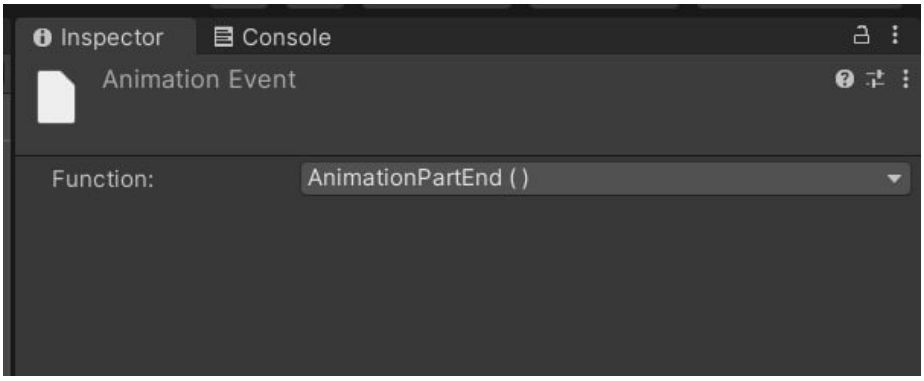


9. Запускаємо сцену і бачимо, що анімація програється.

10. Але для того, щоб обмежити кількість повторень кадрів, потрібно написати скрипт. Наприклад, обмежити рух об’єкта :








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

public class AnimationController : MonoBehaviour
{
        public int animationReplyCount = 3;

        public Animator animator;

        private void Start()
        {

            animator = GetComponent<Animator>();
        }

        public void AnimationPartEnd()
        {

            animationReplyCount--;
            Debug.Log(animationReplyCount);
            if (animationReplyCount <= 0)
            {
                animator.Play("Anim");
            }
        }
    }
```

Завдання 2. З Unity Магазину (Asset Store) завантажити і імпортувати безпосередньо у власний проект анімації запропонованого об’єкта. Обрати 5-6 анімацій і застосувати до головного герою.

Методичні рекомендації

Покроковий опис:

1. Створюємо 2D-проект.
2. Створюємо папки Sprites, Scripts та Scenes (рис. 1).

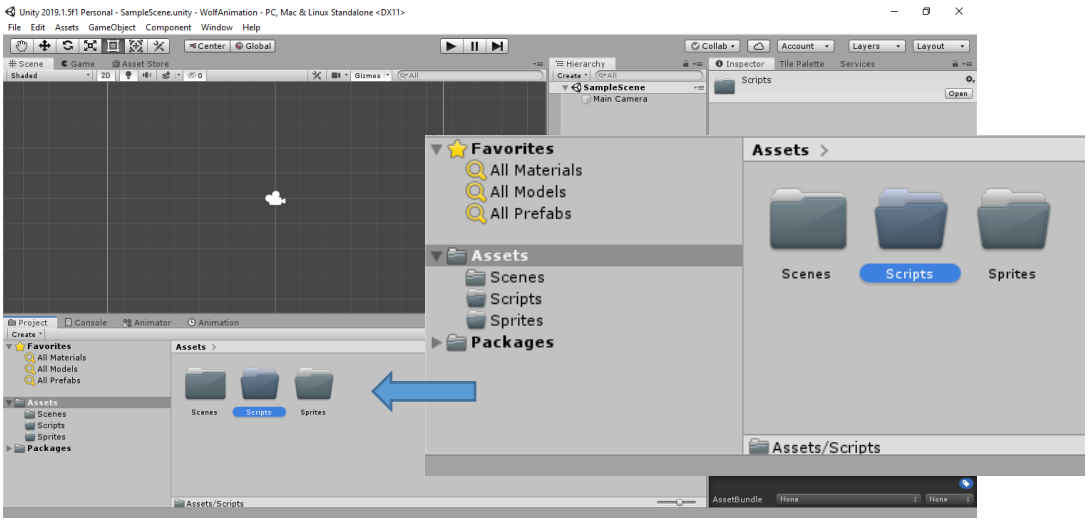


Рис. 1. Створення необхідних папок

3. Потім потрібно скачати обраний сет з Asset Store, в даному прикладі - «Вовка». Маємо два варіанти виконання:

1) В браузері відкрити Asset Store і знайти набір. Додати його собі та відкрити в Unity (рис. 2).

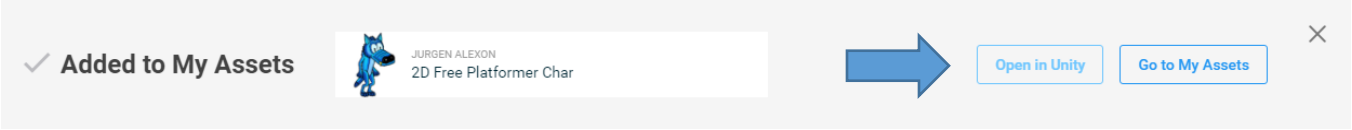


Рис. 2. Додавання за допомогою браузера

2) В Unity відкрити вкладку Asset Store і знайти набір там (рис. 3).

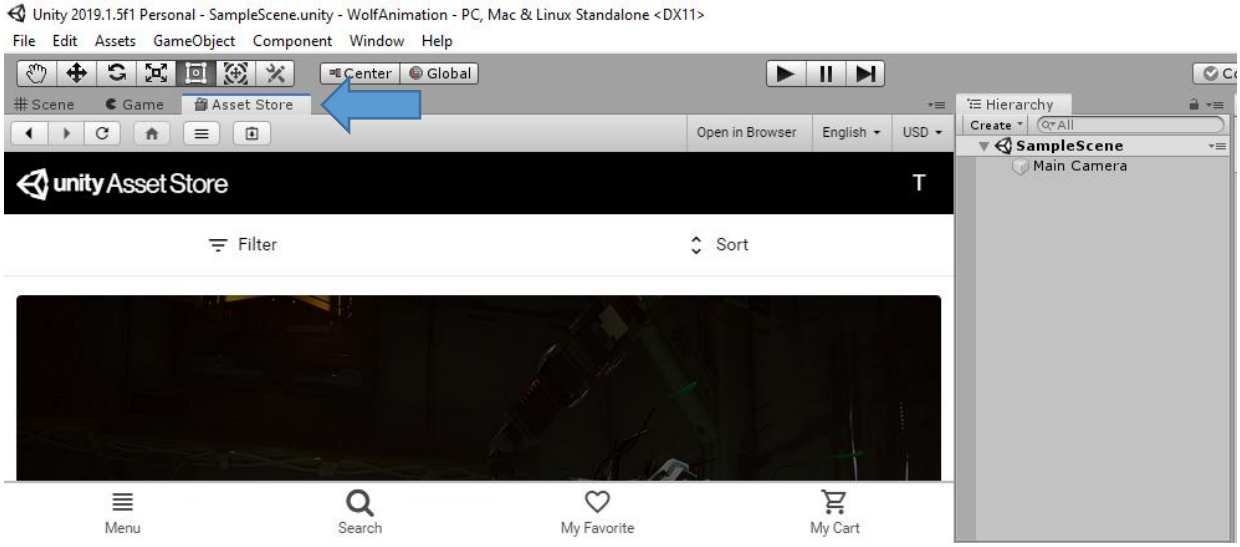


Рис. 3. Asset Store в Unity

При будь-якому обраному варіанті далі необхідно скачати та імпортувати сет в Unity. Для цього у вкладці Asset Store в Unity вибираємо сет та натискаємо Download (рис. 4). Потім Import (рис. 5). Далі обираємо необхідні елементи з набору. У нашому випадку імпортуємо всі (рис. 6).

Рис. 4. Завантаження сету

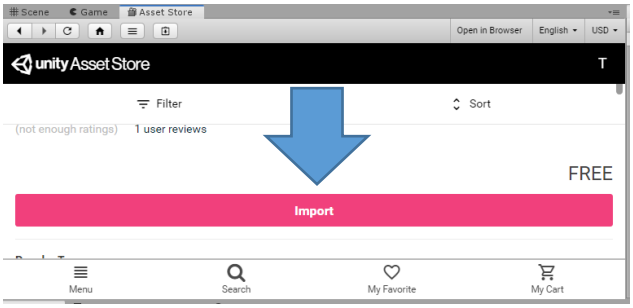


Рис. 5. Імпортування. Етап 1

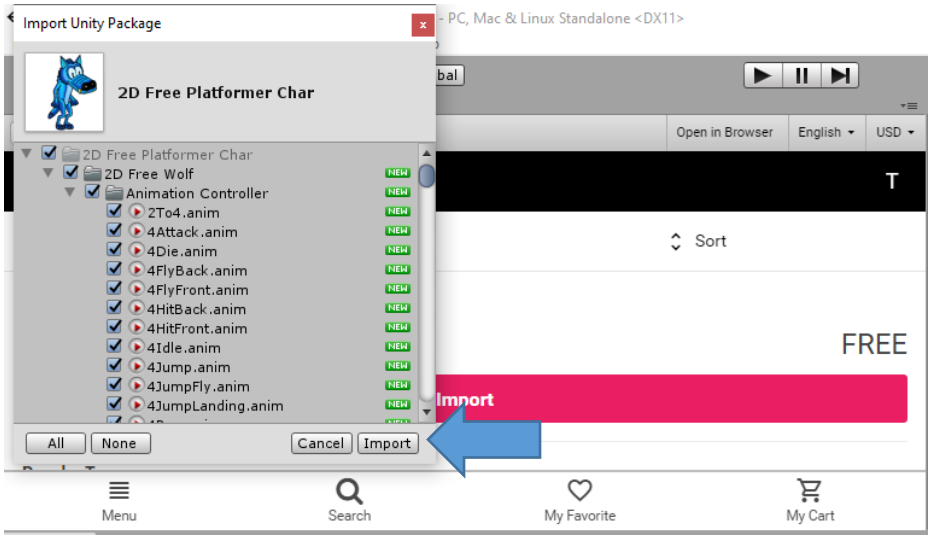


Рис. 6. Імпортування. Етап 2.

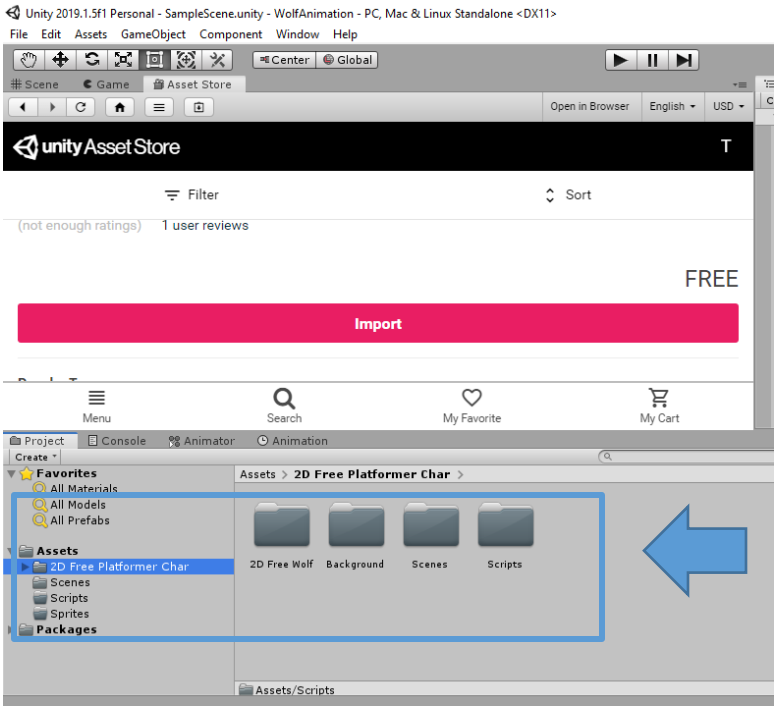


Рис. 7. Імпортовані файли

4. Розміщуємо на сцені фон та платформу. Для цього просто знаходимо їх в папці *2D Free Platformer Char / Background* та перетягуємо або на сцену, або в ієрархію. Персонажа знаходимо в спрайтах: папка *Idle1*, перше зображення. Щоб змінити розташування можна змінювати значення в *position* або обравши другий інструмент на панелі (рис. 8).

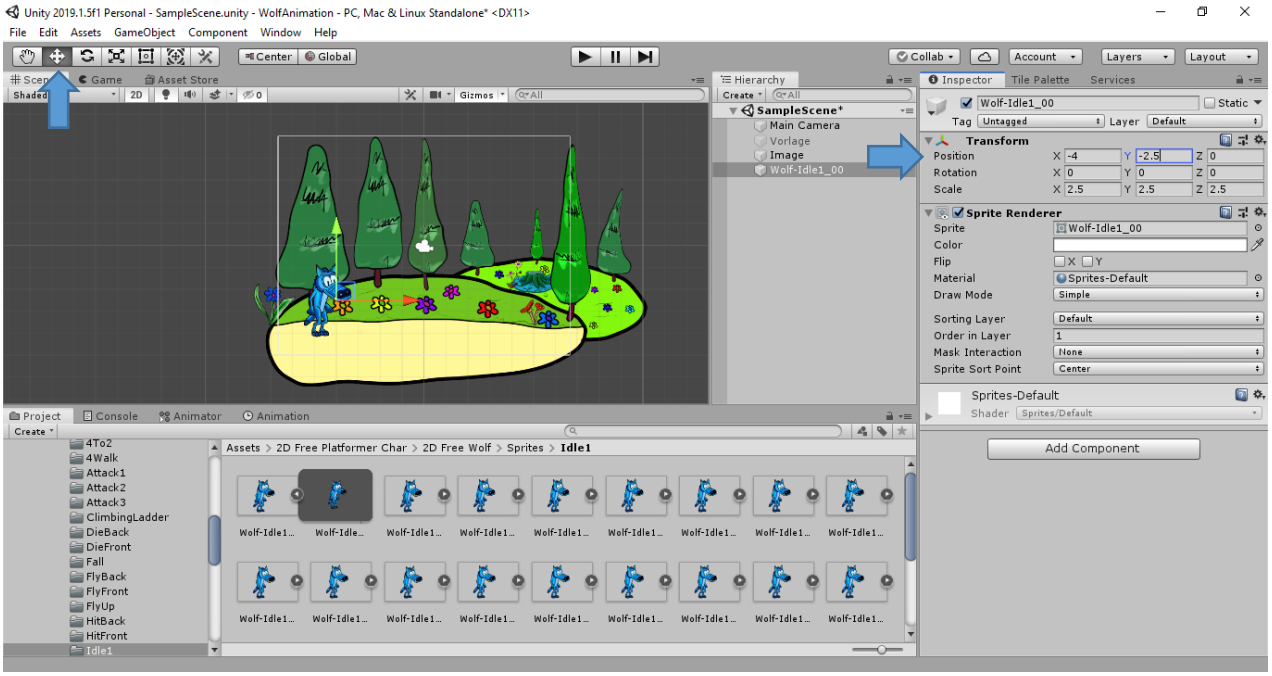


Рис. 8. Розміщення об’єктів на сцені

5. Задаємо необхідні параметри для персонажу та платформи. Почнемо з платформи.

Для того, щоб персонаж не падав вниз при старті гри, платформі необхідно задати колайдер. Тоді він буде на ній стояти. Для цього створюємо новий *2D Object* в ієрархії на цей об'єкт, в інспекторі спускаємось донизу та натискаємо *Add Component*. Обираємо *BoxCollider2D*.

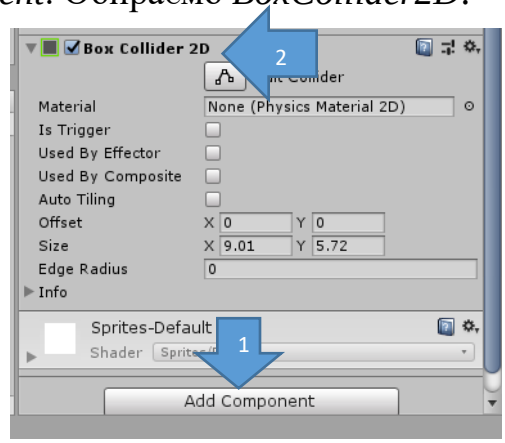


Рис. 9. Колайдер для платформи

Після цього бачимо, що на сцені з'явився зелений прямокутник. Це і є область колайдера (рис. 10). Але квадрат замалий. Тому натискаємо Rect tool та вирівнюємо до потрібних розмірів. Та для того щоб платформа не відображалась знімемо галочку з компонента *Sprite Renderer* (рис. 11).

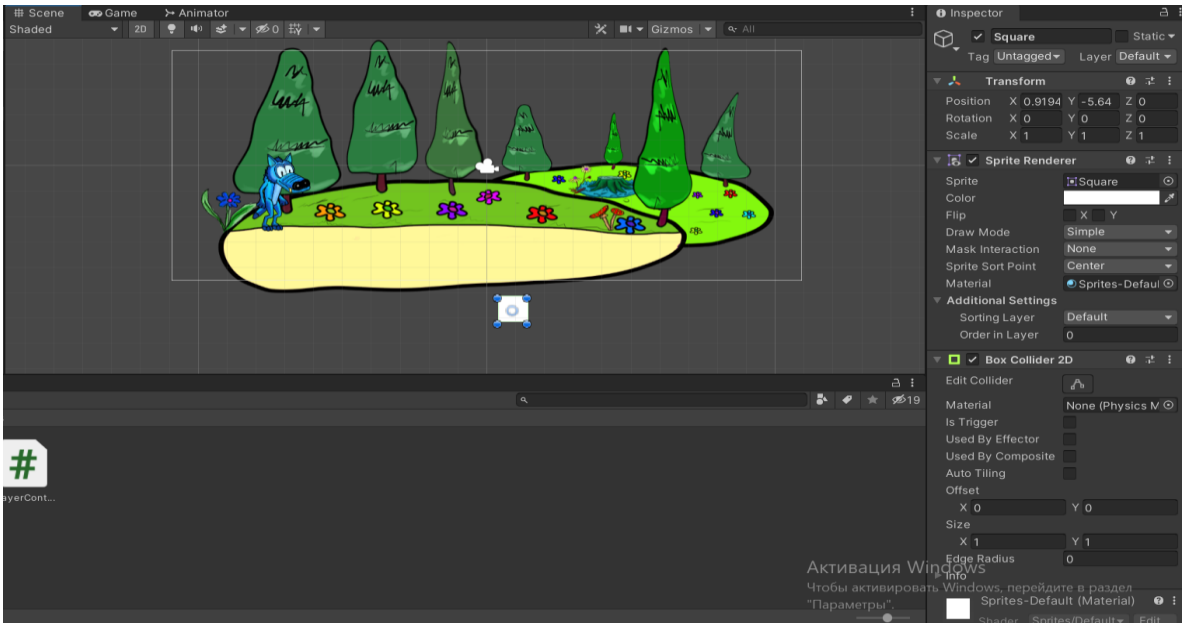


Рис. 10. Створений колайдер

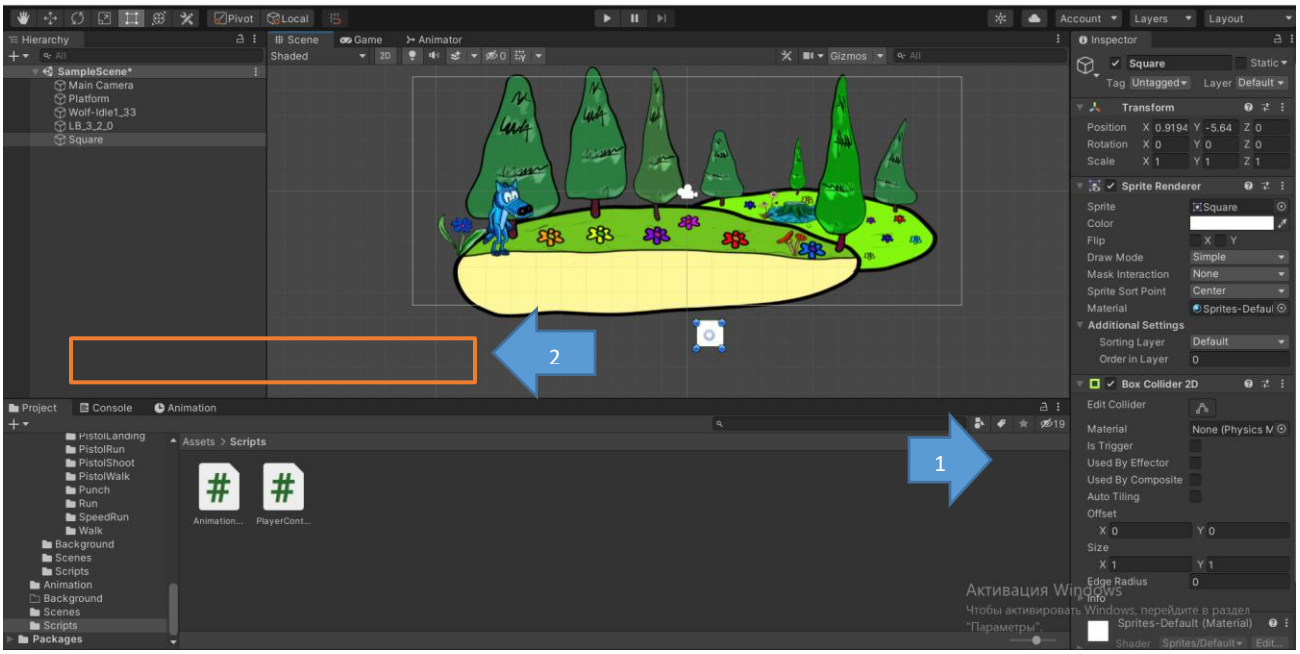


Рис. 11. Зміна розміру колайдера

Тепер працюємо з об'єктом персонажу. В ієрархії виділяємо Wolf (персонажа). В інспекторі натискаємо *Add Component* та обираємо *CapsuleCollider2D*. За допомогою *Edit Collider* налаштовуємо його форму (рис. 12).

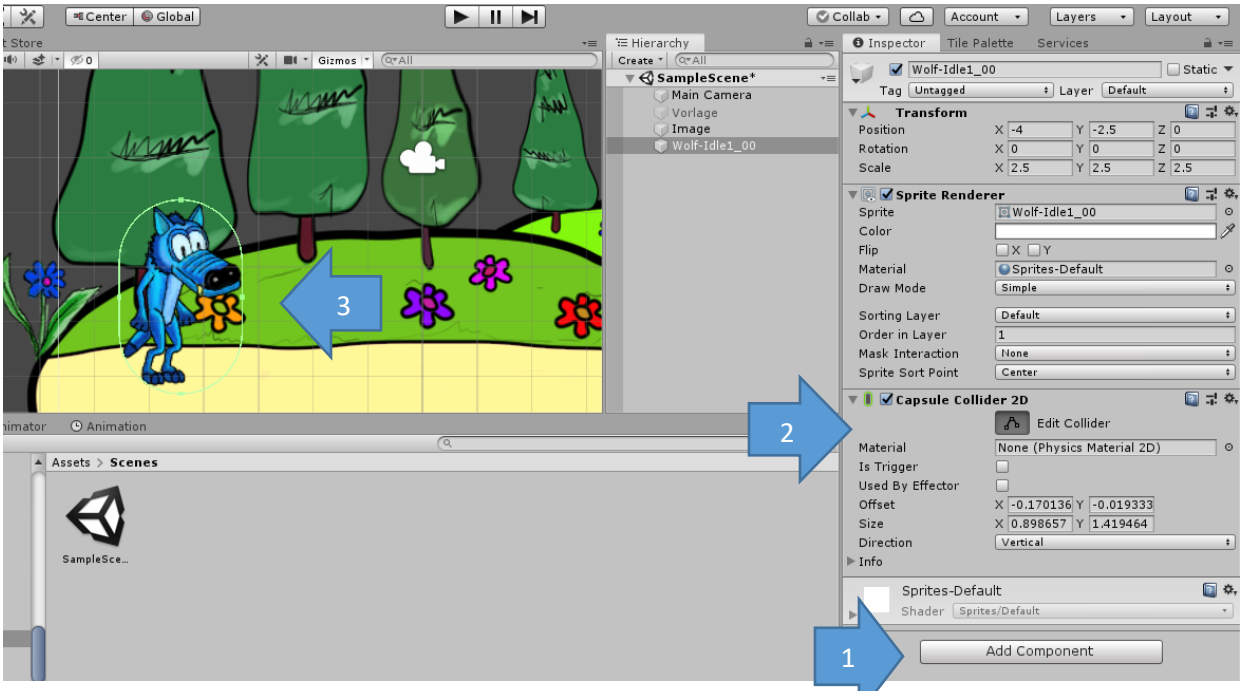


Рис. 12. Колайдер для персонажу

Далі необхідно задати йому фізичні властивості. Для цього натискаємо *Add Component – Rigitbody2D* (рис. 13).

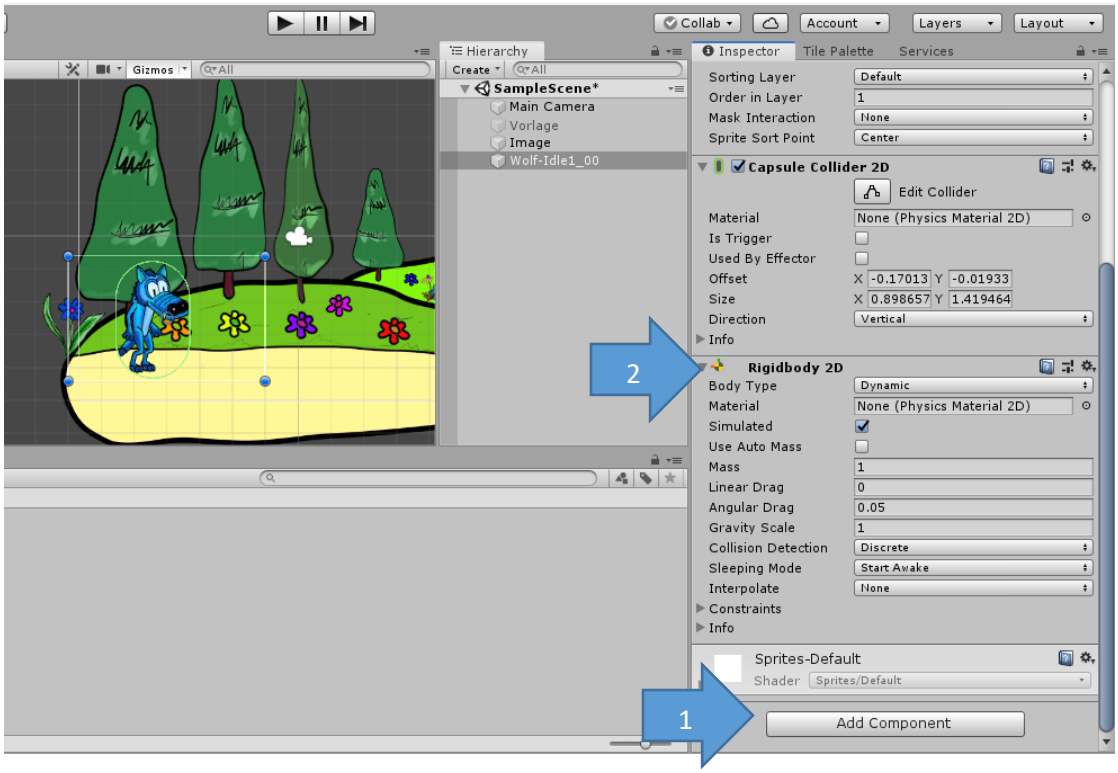


Рис. 13. Додаємо Rigidbody для персонажу

Для нормальної ходьби персонажу, обов'язково необхідно заборонити йому зміни координат по z (рис. 14) та в *Layer* встановити *Ignore Raycast* (рис. 15). Останнє необхідно для того, щоб персонаж реагував на землю. Це знадобиться при написанні коду для стрибка. В *Rigidbody Gravity Scale* встановіть 2.

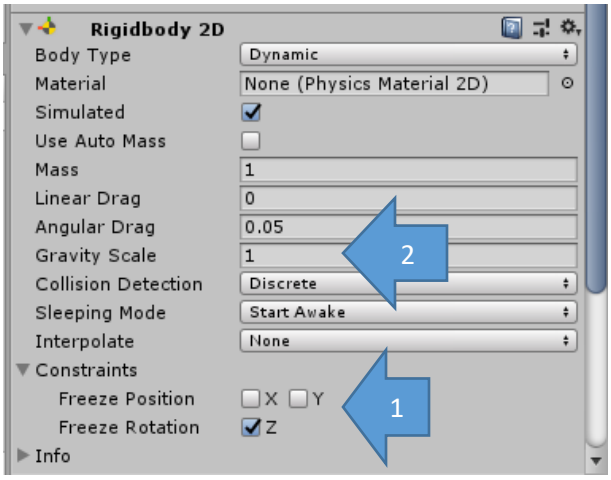


Рис. 14. Додаткові налаштування

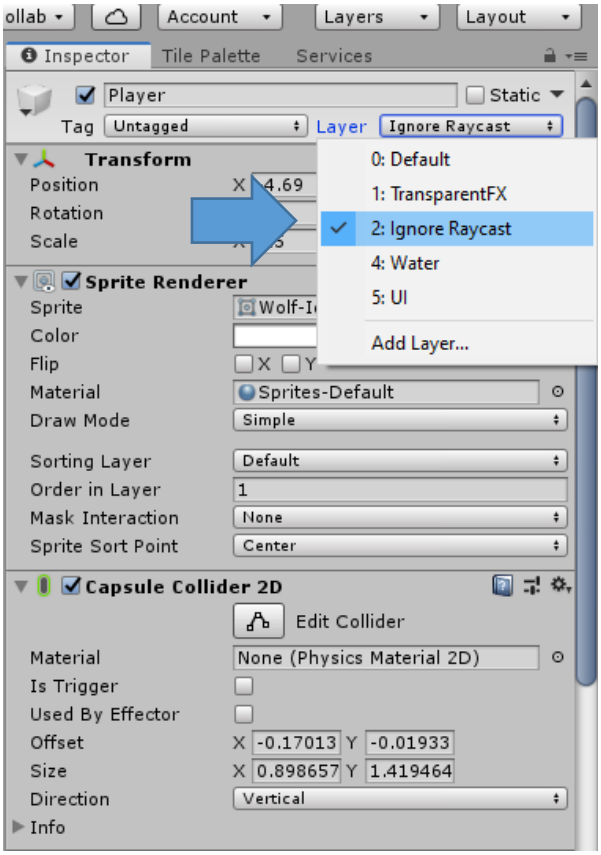


Рис. 15. Додаткові налаштування

Останнім до персонажа потрібно прикрипити аніматор. *Add Component – Animator* (рис. 16). Це потрібно для взаємодії персонажа з анімаціями.

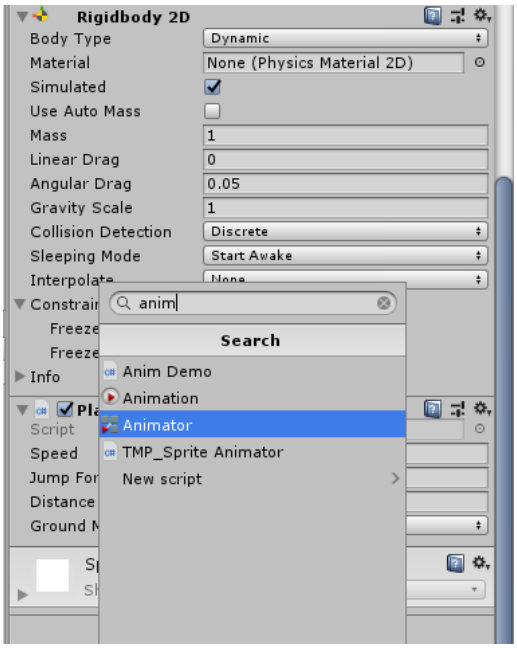


Рис. 16. Додаємо аніматор

6. Наступним кроком обираємо анімації, які ми будемо використовувати. В даному випадку – *Idle1*, *Run*, *Jump*, *Landing*, *Kick*, *Punch*. Нам необхідно набір спрайтів перетворити в анімацію. Для цього заходимо в папку зі спрайтами і всі їх виділяємо (рис. 17).

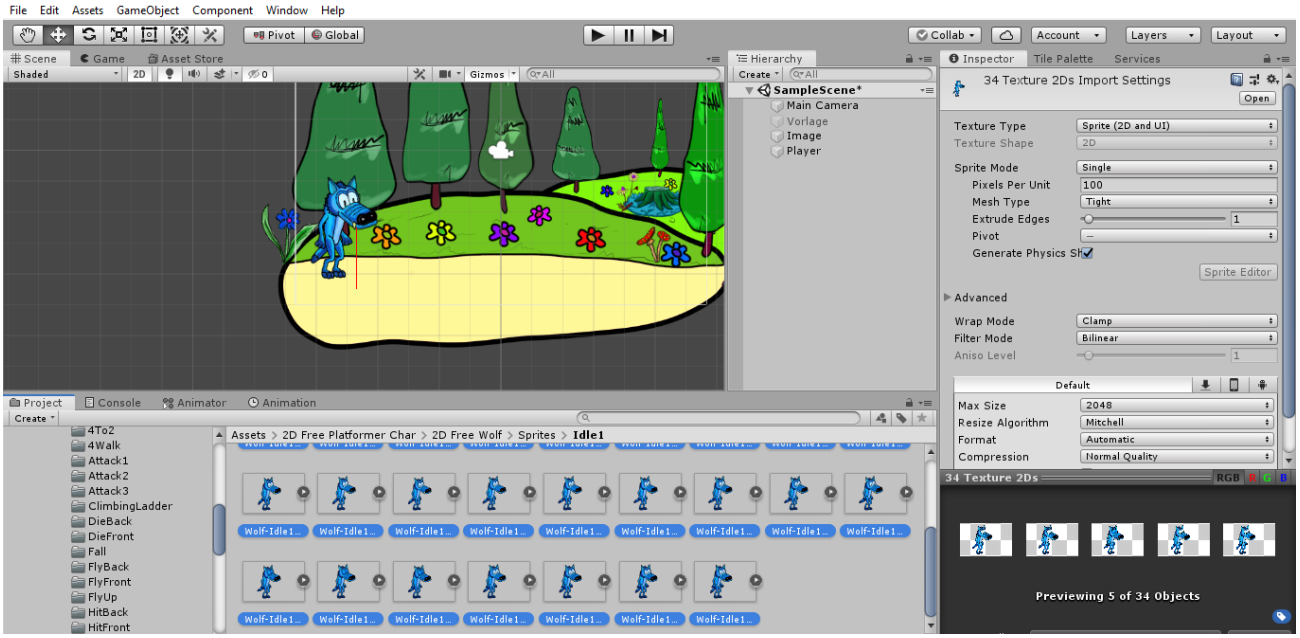


Рис. 17. Виділення спрайтів

Затискаємо ліву кнопку миші й перетягуємо їх на персонажа (*Player*). Вам *Unity* запропонує зберегти анімацію. Потрібно зайти в папку *Assets* та створити нову папку *Animations*. Туди будемо зберігати всі подальші анімації (рис. 18).

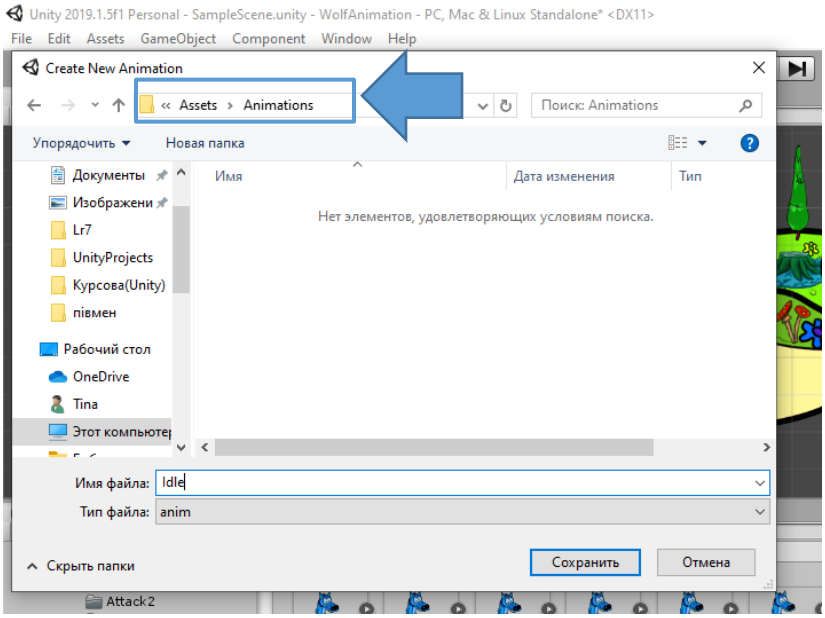


Рис. 18. Збереження анімації

Далі переходимо на вкладку Animation. Важливо, щоб при цьому був виділений персонаж в ієрархії. Тут ми можемо продивитись анімацію (кнопка play), записувати чи змінювати її. в нашому випадку потрібно лише підібрати оптимальну частоту кадрів. Це значення потрібно змінювати в *Samples*. Для першої анімації обираємо 34 і натискаємо Enter. Хоча це значення можна змінювати за власним смаком (рис. 19).

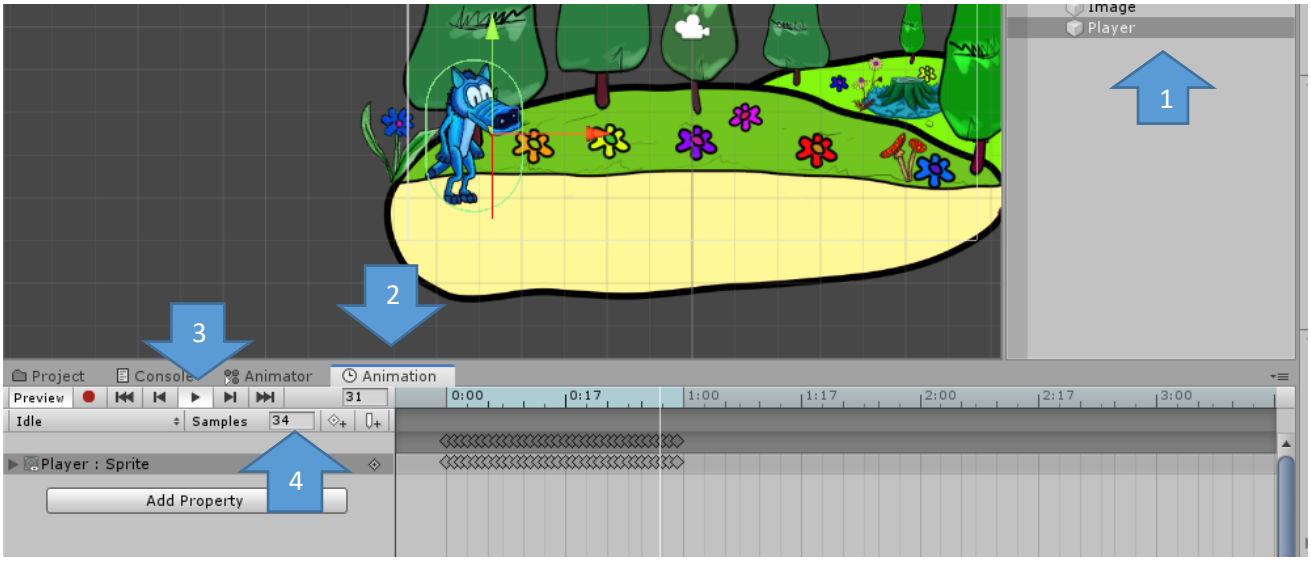


Рис. 19. Обробка анімації

Такі ж дії виконуємо для інших анімацій. Щоб працювати з іншою анімацією необхідно обрати її в списку (рис. 20), але вже після додавання її на персонажа.

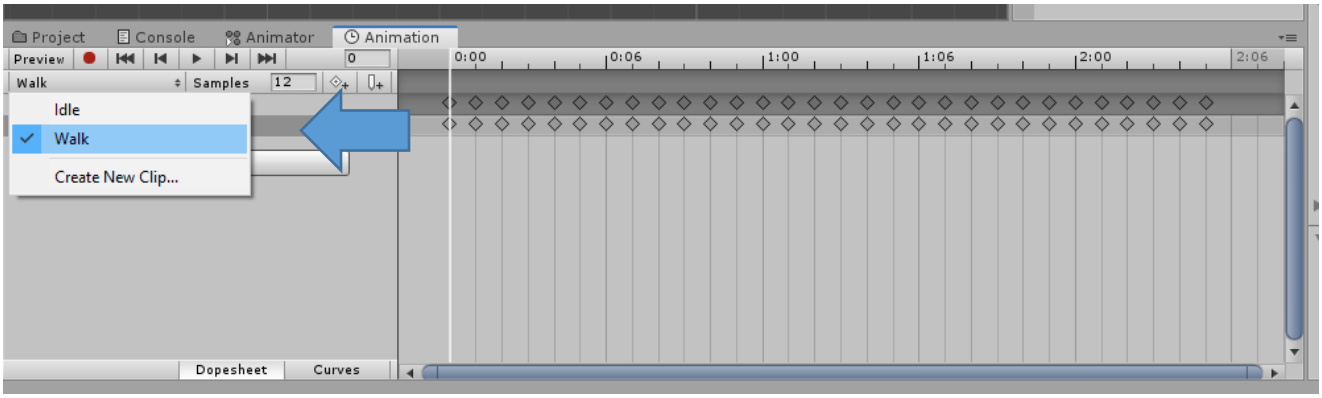


Рис. 20. Вибір іншої анімації

Ще для деяких анімацій необхідно вимкнути повторення. Для цього знаходимо у скачаному сеті анімацію, натискаємо на неї і в інспекторі знімаємо галочку *Loop Time* (рис. 21). У нашому випадку непотрібно повторення *Jump, Landing, Kick і Punch*.

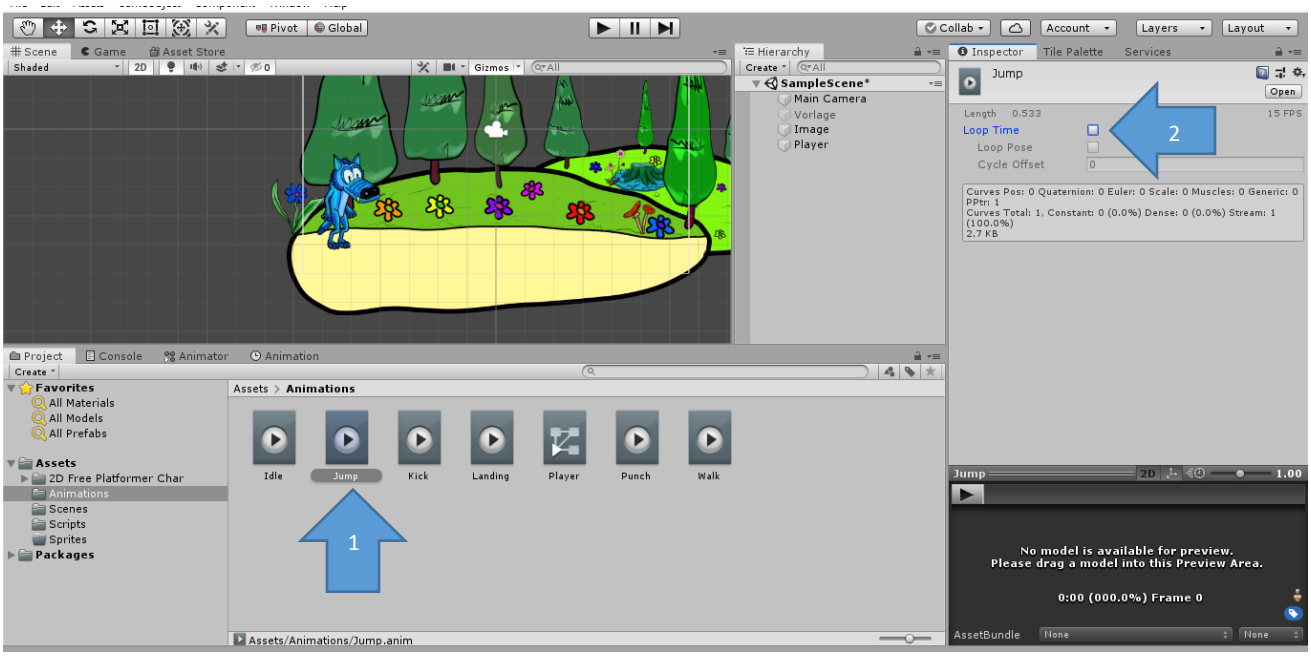


Рис. 21. Додаткові налаштування для анімації

7. Отже, ми додали анімації на персонажа. Тепер нам необхідно контролювати переходи між ними. Для цього переходимо у вкладку *Animator* (рис. 22).

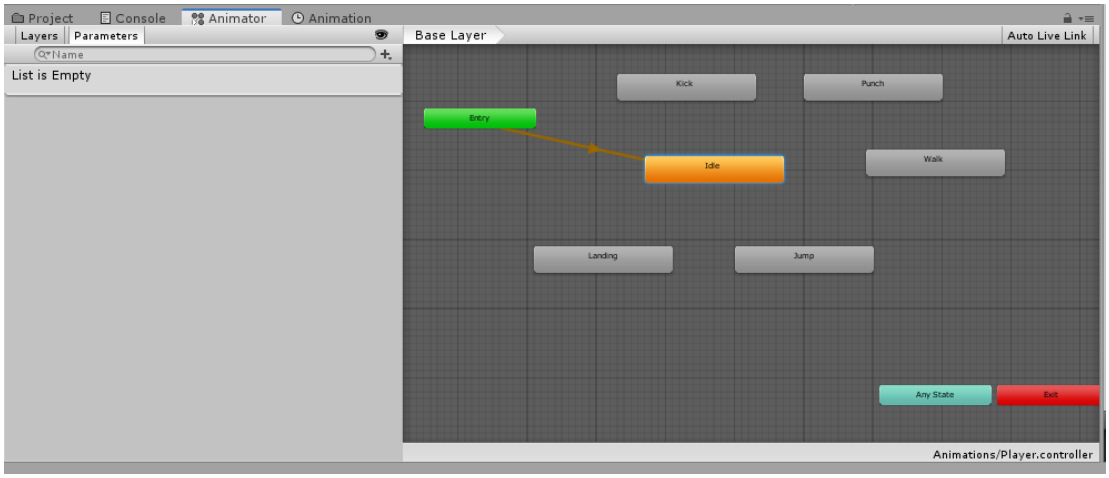


Рис. 22. Аніматор

Тут ми бачимо всі наявні анімації. Намагайтесь розташовувати їх впорядковано. Помаранчева стрілка позначає анімацію по замовчуванню. Натискаємо на *Idle* правою кнопкою миші, обираємо *MakeTransition* та наводимо курсор на Walk. Так само робимо з Walk до *Idle*. Ми створили зв'язок між двома анімаціями. Тепер потрібно задати умову (чи умови) при якій буде відбуватись перехід з одної анімації на іншу. У вкладці *Parameters* натискаємо «+» та обираємо *Float* (рис. 23). Назвемо його Move.

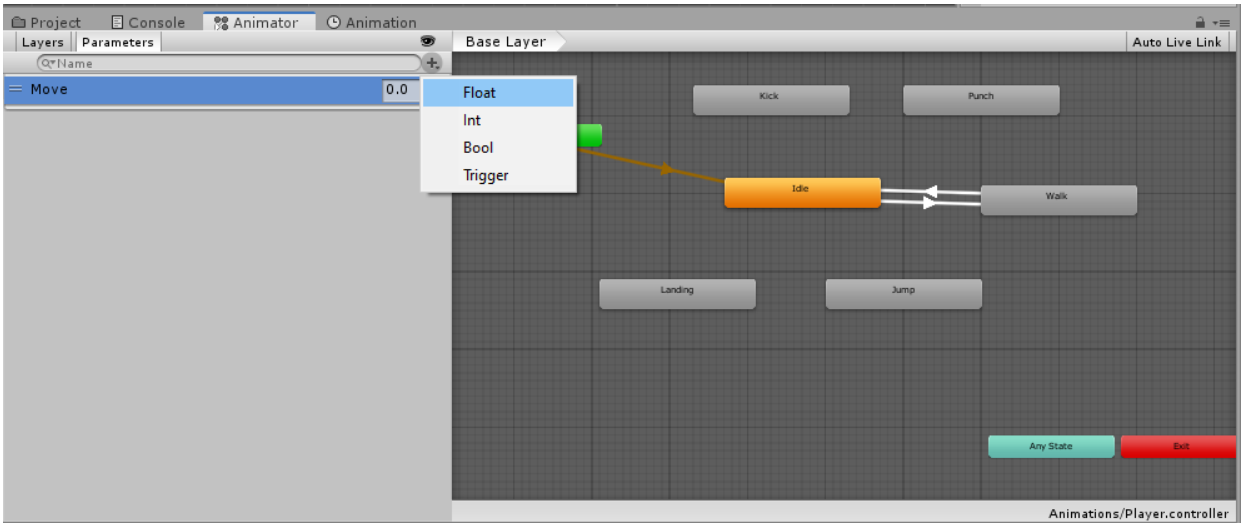


Рис. 23. Задаємо параметр для анімації

Натискаємо на стрілку переходу з Walk в *Idle*. Вона має бути блакитного кольору. В інспекторі знімаємо галочку *Has Exit Time*, зменшуємо час, додаємо умову *(Conditions) Move Greater 0.1* (рис. 24). Для чого це все необхідно? Коли ми будемо рухатись в коді змінна буде приймати число, відмінне від нуля. Тобто ми починаємо рух і анімація повинна перемкнутись на Walk. Для того, щоб під час руху увесь час програвалась анімація вимикаємо *Exit Time*.

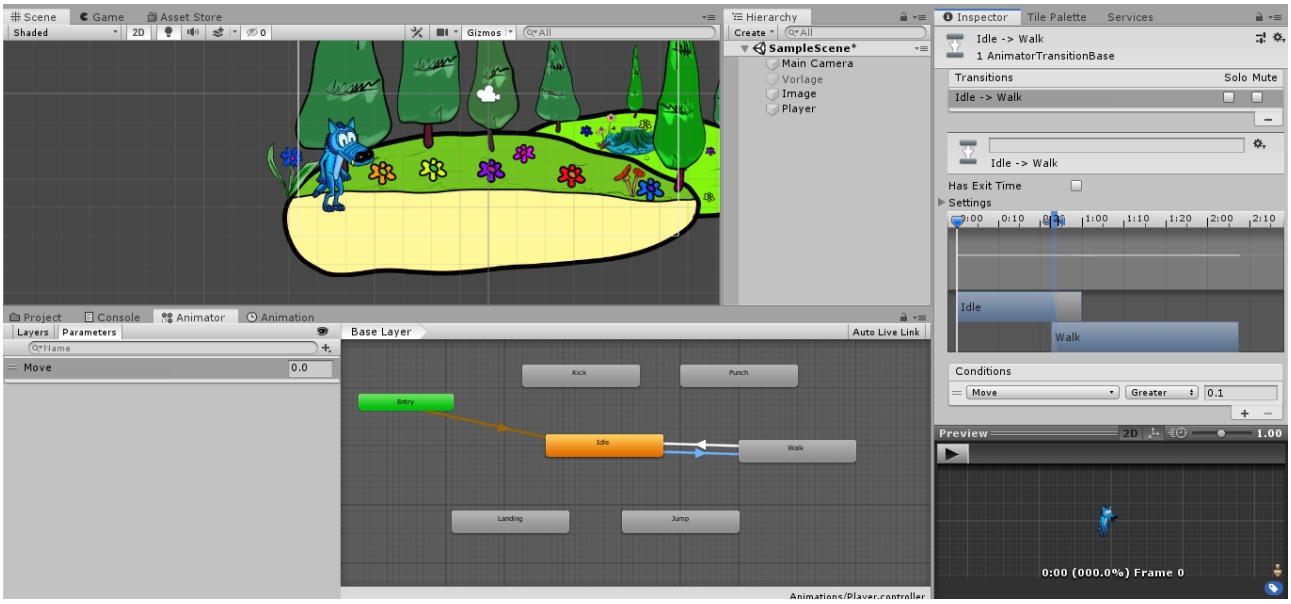


Рис. 24. Налаштування для Idle to Walk

Далі йде опис для всіх анімацій в рисунках 25 - 39. Звертайте увагу на напрямок стрілок, умови (*Conditions*) та налаштування часу переходу (*Settings*).

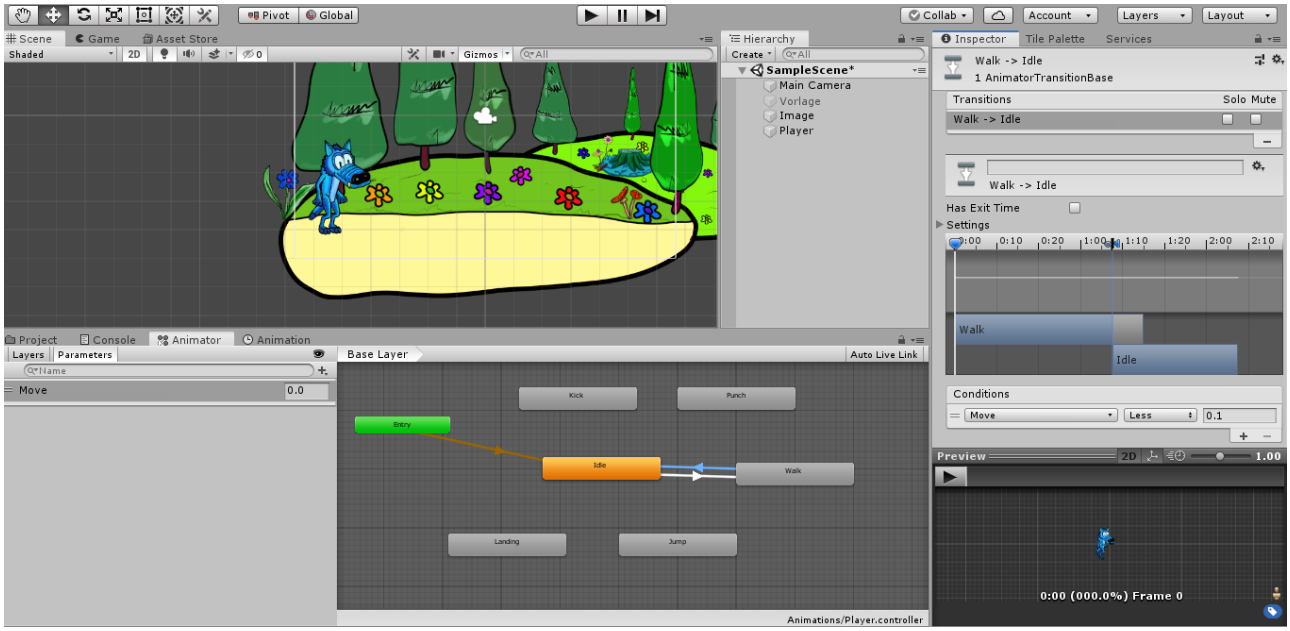


Рис. 25. Налаштування для Walk to Idle

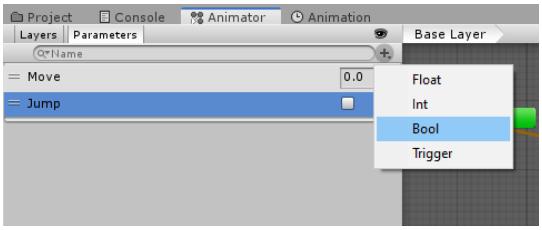


Рис. 26. Додаємо параметр Jump

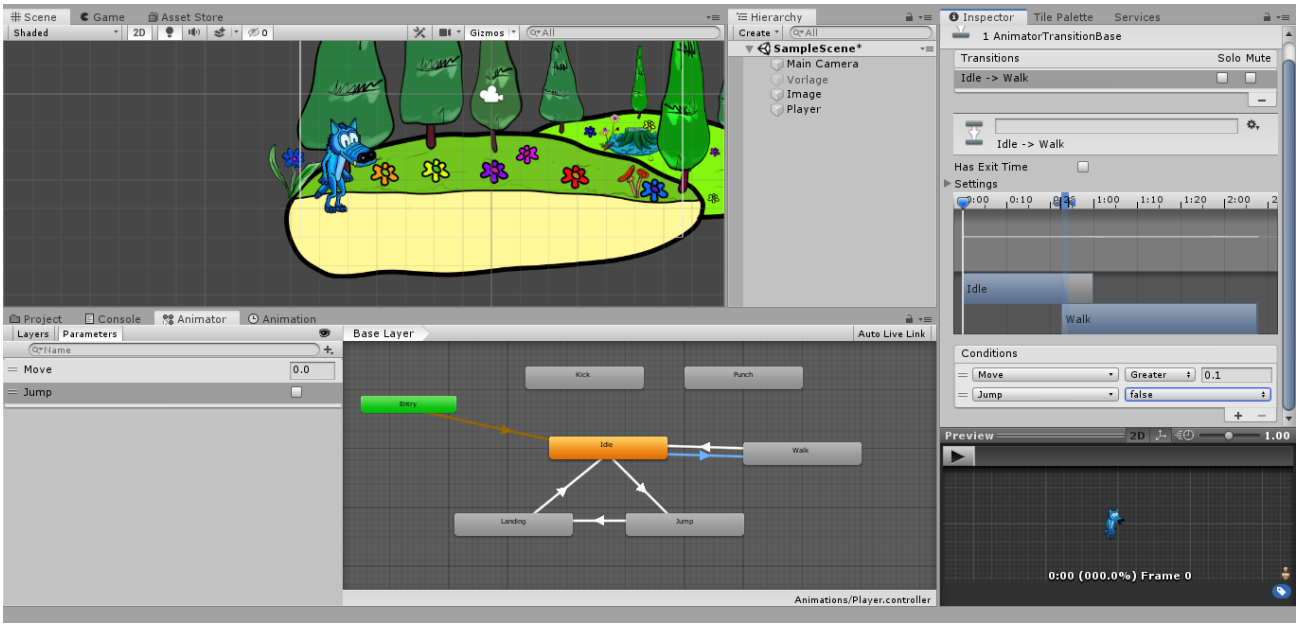


Рис. 27. Додаємо параметр Jump для Idle to Walk

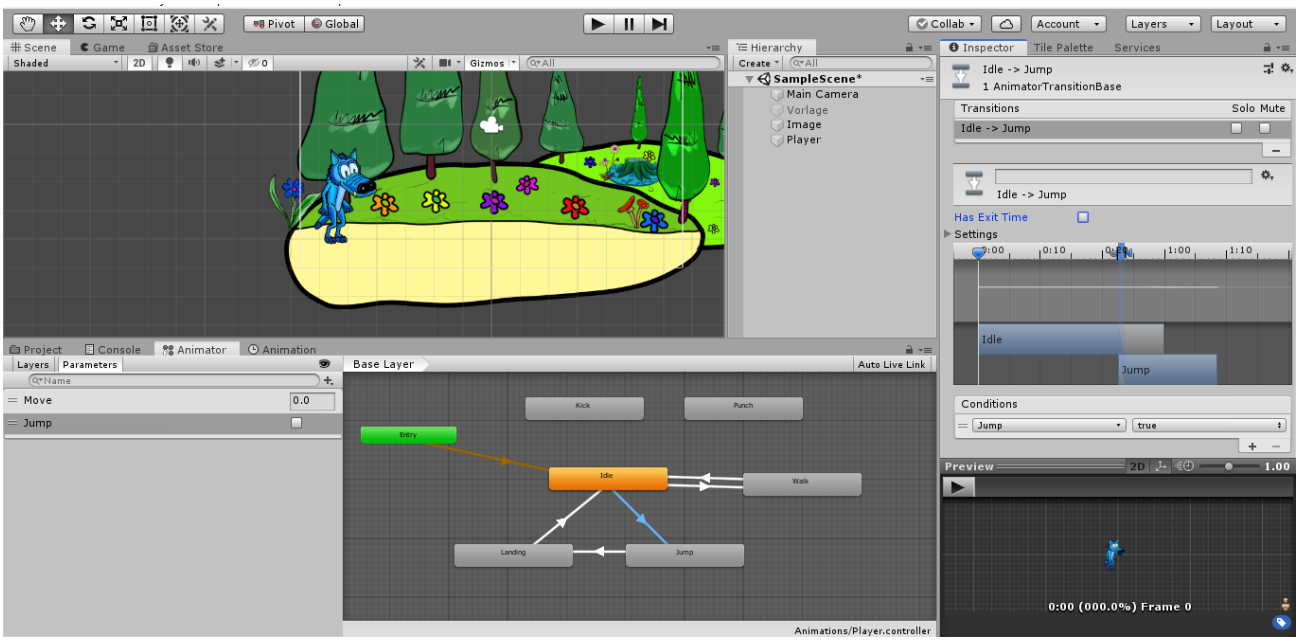


Рис. 28. Налаштування для Idle to Jump

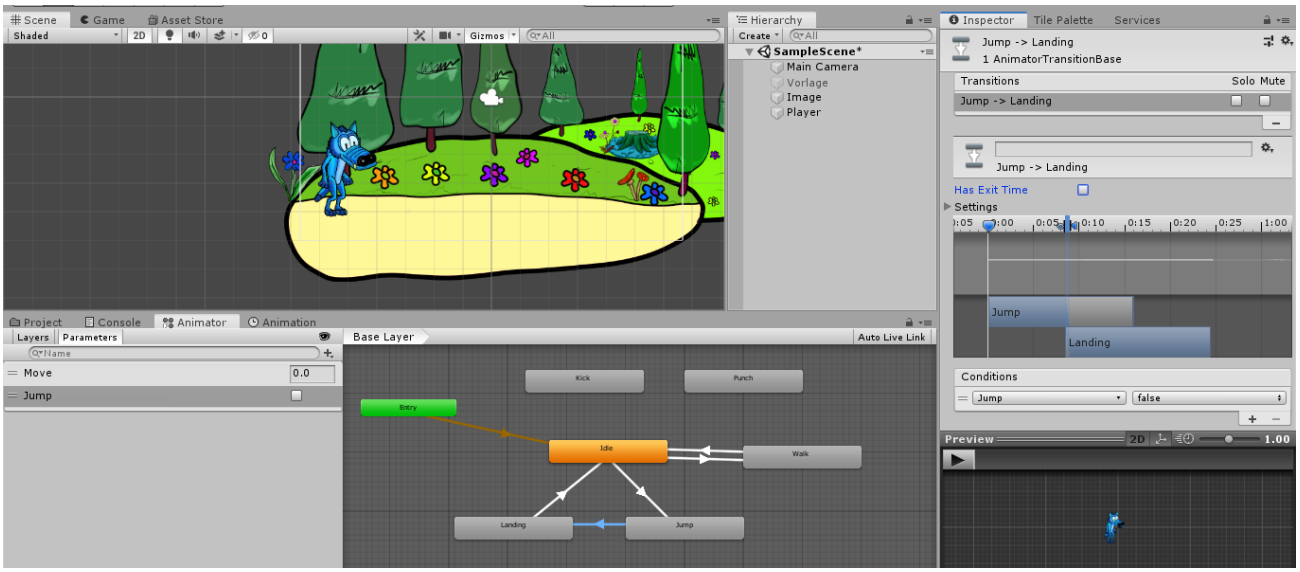


Рис. 29. Налаштування для Jump to Landing

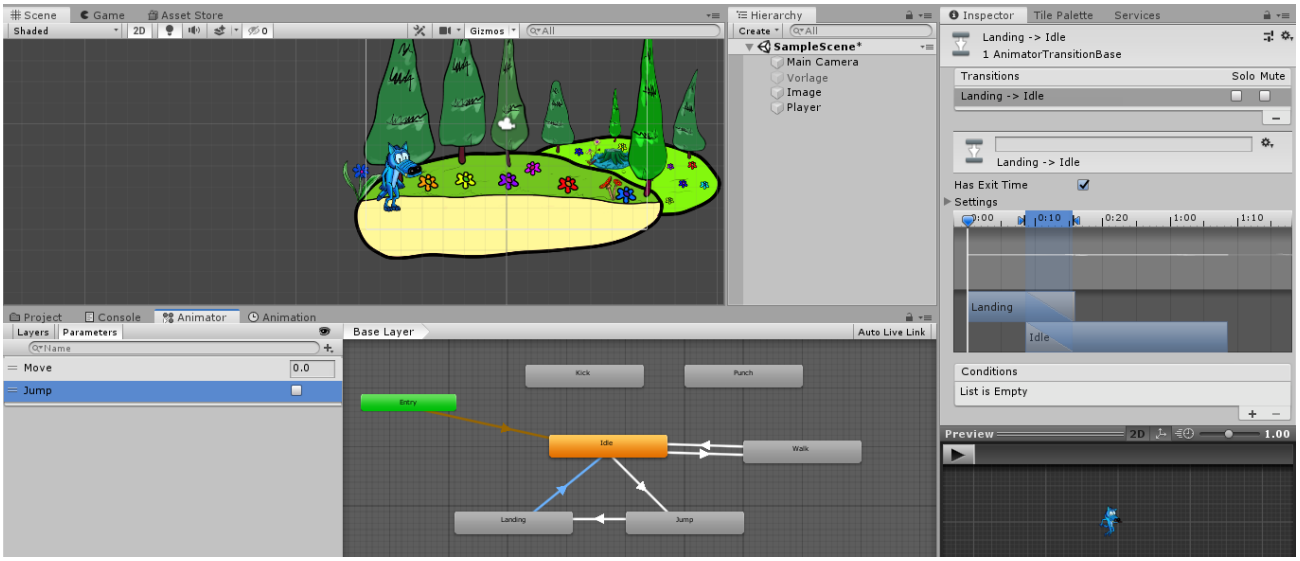


Рис. 30. Налаштування для Landing to Idle

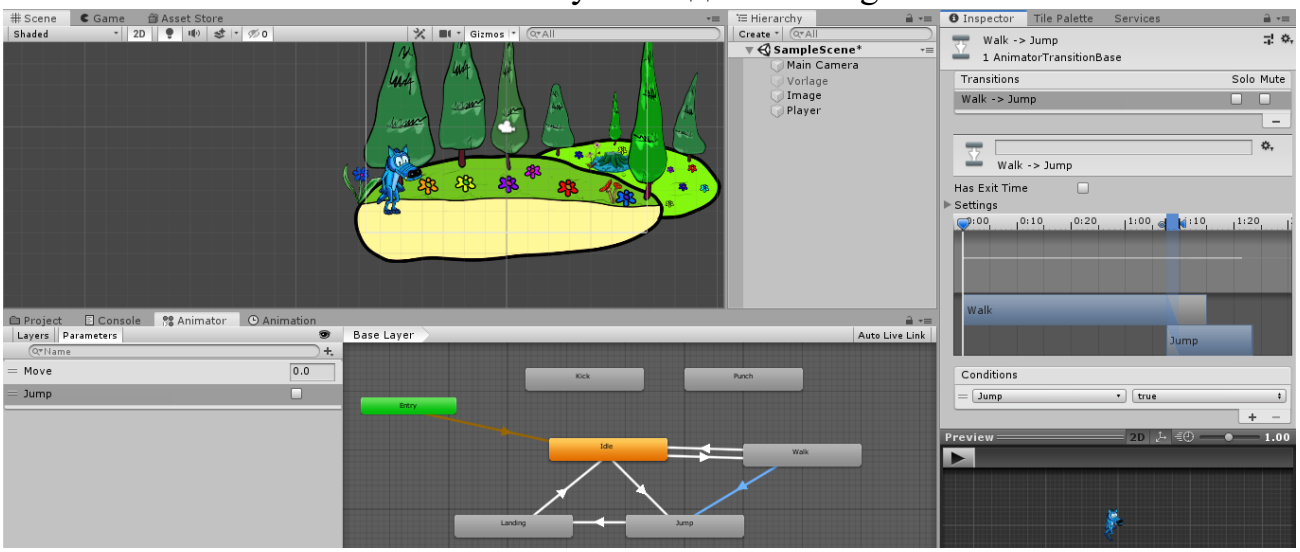


Рис. 31. Налаштування для Walk to Jump

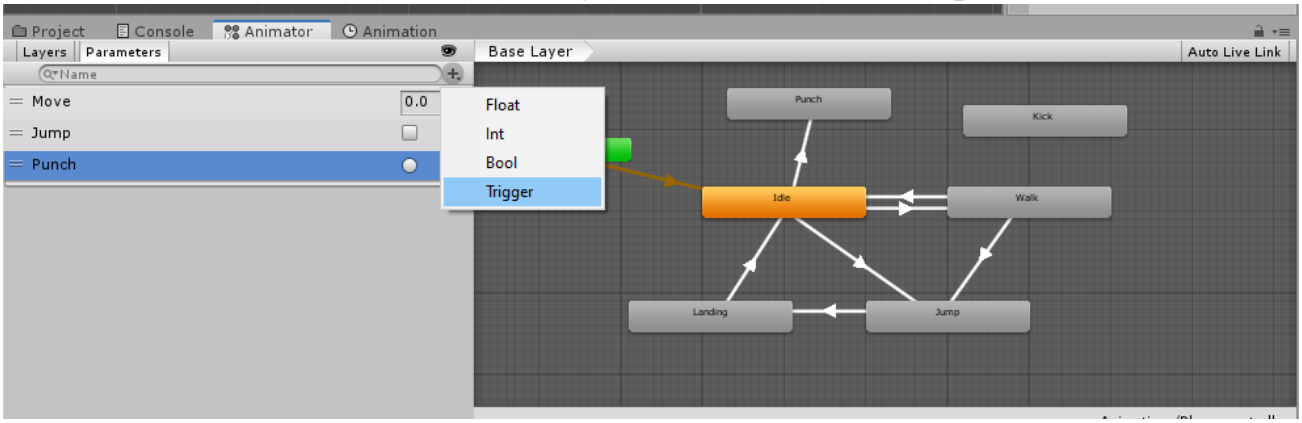


Рис. 32. Додаємо тригер Punch

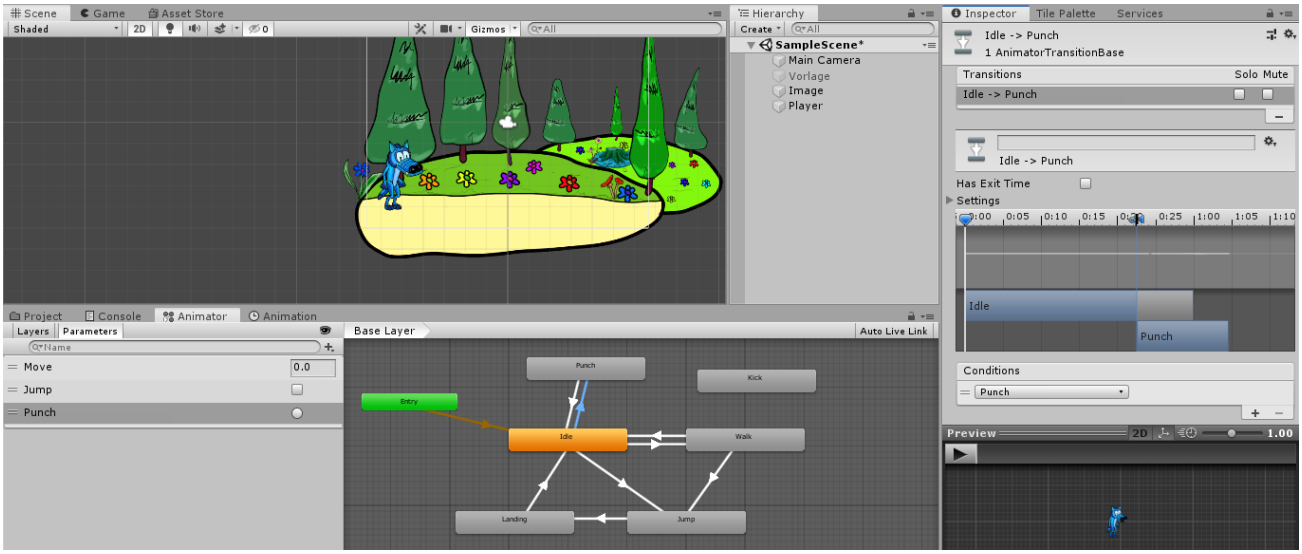


Рис. 33. Налаштування для Idle to Punch

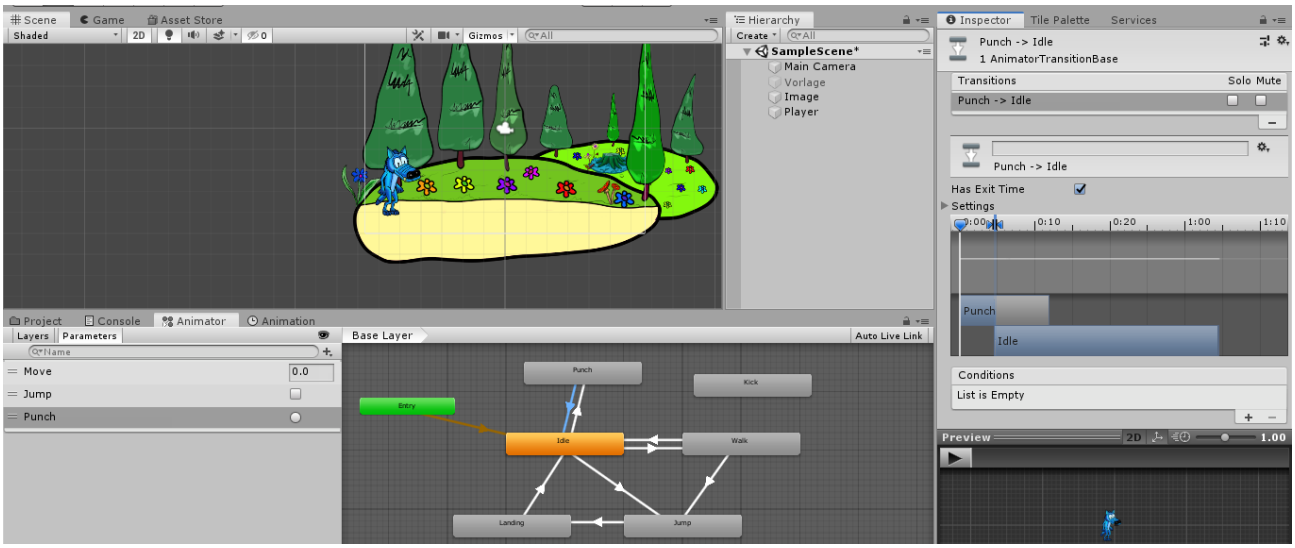


Рис. 34. Налаштування для Punch to Idle

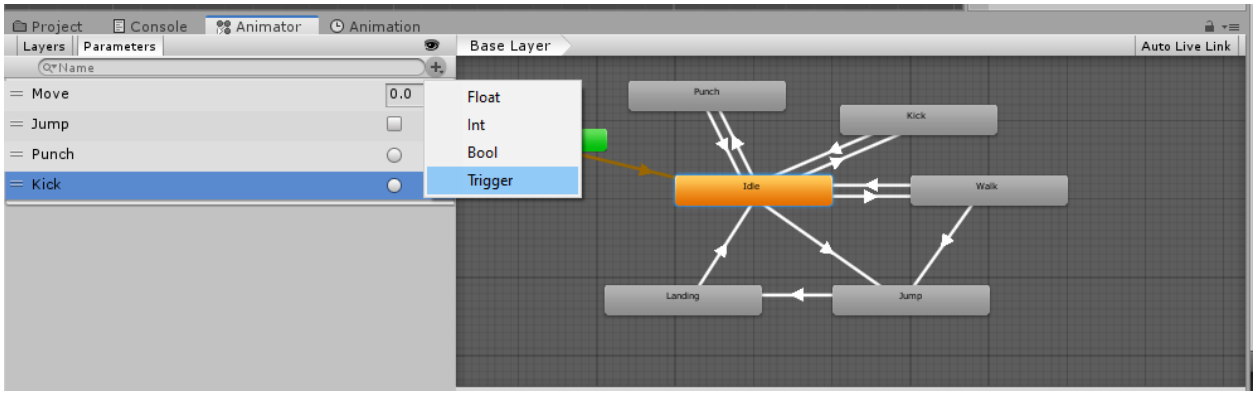


Рис. 35. Додаємо тригер Kick

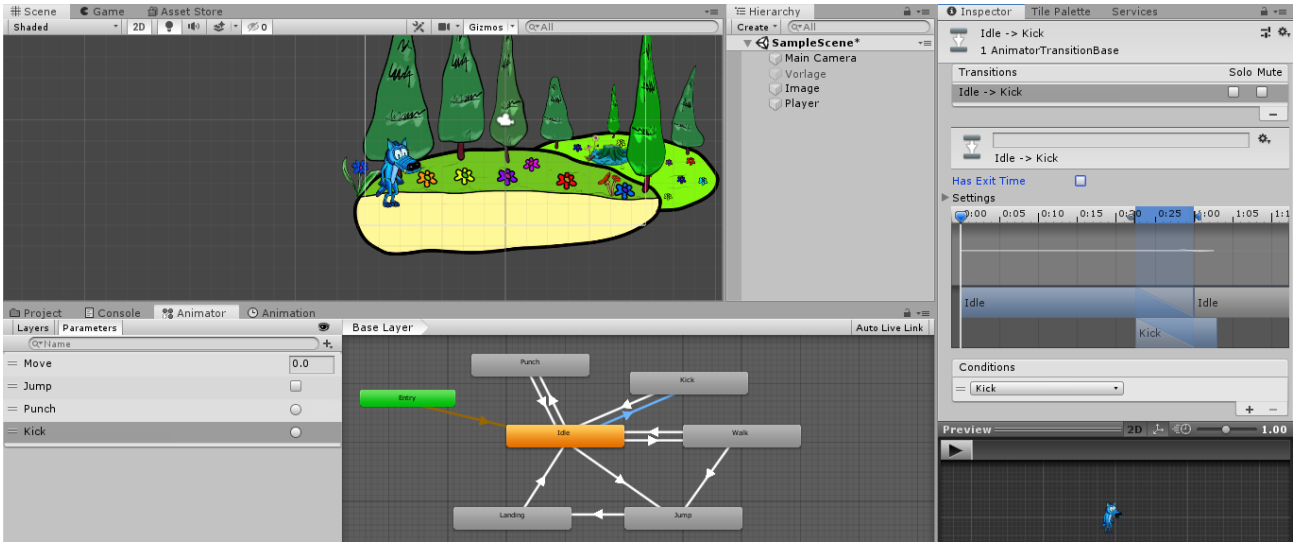


Рис. 36 Налаштування для Idle to Kick

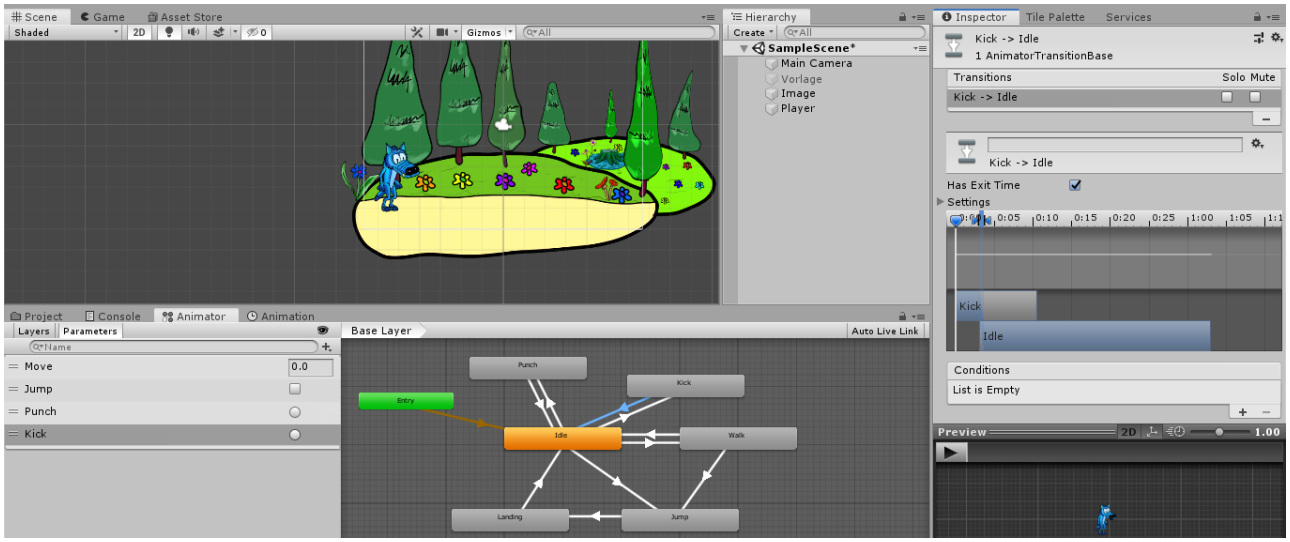


Рис. 37 Налаштування для Lick to Idle

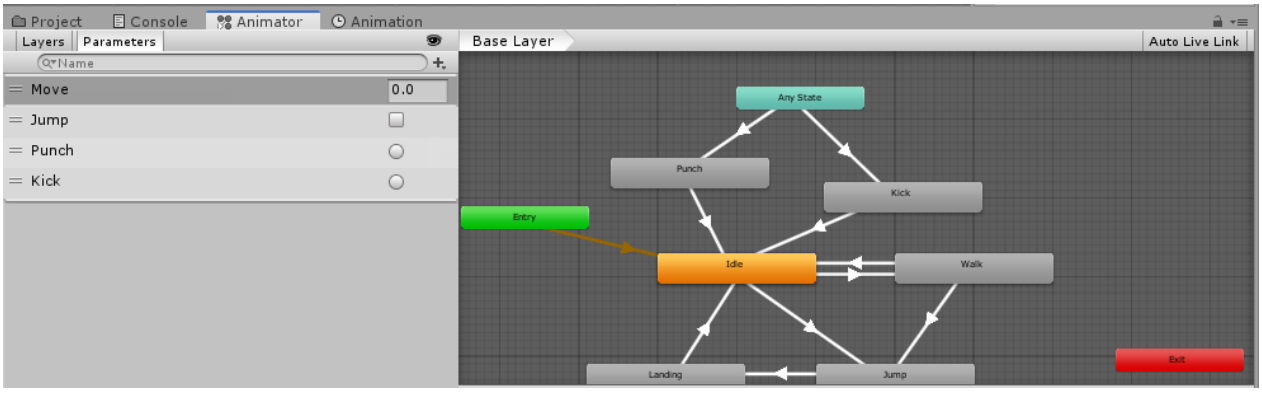


Рис. 38. Додаємо взаємодію з Any State

/ Стрілки від Idle до Punch та від Idle до Kick, як на рисунку 39 додавати не потрібно!!!*/*

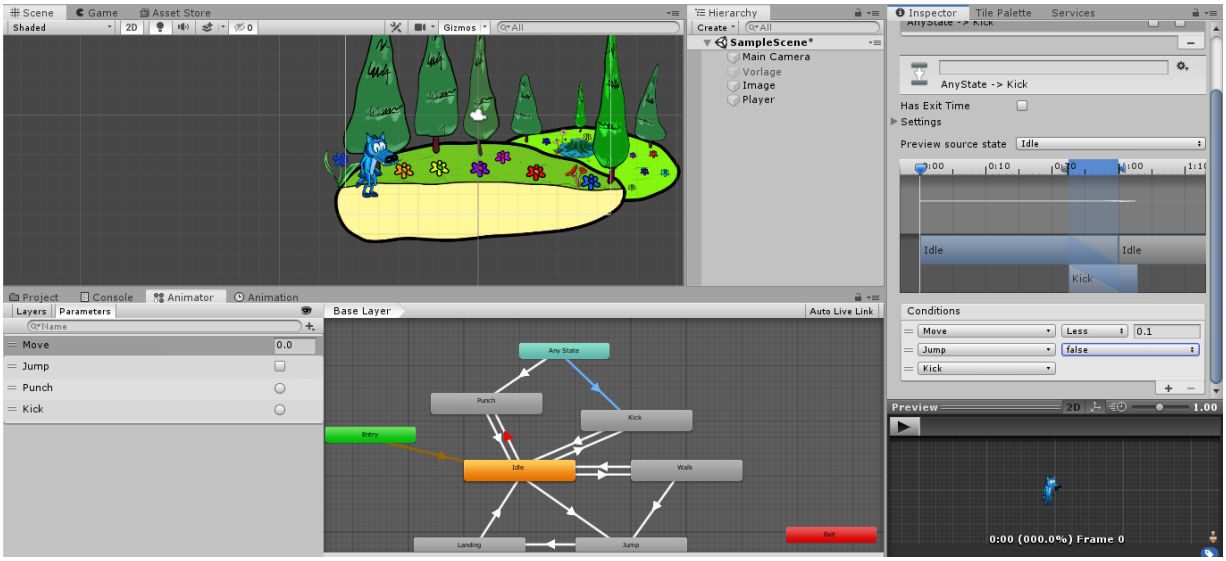


Рис. 39. Налаштування для Any State to Kick та Any State to Punch

8. Залишилось написати C# Script, який буде все це контролювати. Створюємо в папці Scripts новий скрипт.

Рис. 40. Створення скрипта

Двічі клацнувши на нього переходимо у Visual Studio.

Лістинг:

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

public class PlayerController : MonoBehaviour
{
    public float speed = 5f;
    public float jumpForce = 2f;
    public float distanceToGround;
    public LayerMask groundMask;
    private Rigidbody2D rb;
```

```
private Animator anim;
private float horizontal;
private bool facingRight;
private bool grounded;

private void Start() {
    rb = GetComponent<Rigidbody2D>();
    anim = GetComponent<Animator>();
}

private void Update() {
    //перевірка чи персонаж стоїть на землі
    CheckGround();
    if (Input.GetKeyDown("space") && grounded)
    {
        Jump();
    }
}

private void FixedUpdate() {
    //вхідні дані про переміщення вліво-вправо
    horizontal = Input.GetAxis("Horizontal");
    //анімація ходьби
    anim.SetFloat("Move", Mathf.Abs(horizontal));
    //зміна напрямку персонажа
    if (horizontal > 0 && facingRight || horizontal < 0 && !facingRight)
        Flip();
    //анімація ударів
    if (Input.GetMouseButtonDown(0))
        anim.SetTrigger("Punch");
    if (Input.GetMouseButtonDown(1))
        anim.SetTrigger("Kick");
    //реалізація стрибку та його анімації
    //анімація приземлення після стрибку
    if (rb.velocity.y < 0)
        anim.SetBool("Jump", false);
    //переміщення персонажа вліво-вправо
    if (horizontal != 0)
        Move();
}

private void CheckGround()
{
    grounded = Physics2D.Raycast(rb.position, Vector3.down, distanceToGround,
    groundMask);
}
private void Jump()
```

```
    {
        rb.AddForce(Vector3.up * jumpForce, ForceMode2D.Impulse);
        anim.SetBool("Jump", true);
    }
    private void Move()
    {
        rb.velocity = new Vector3(horizontal * speed, rb.velocity.y);
        Vector3 position = rb.position;
        position.x = Mathf.Clamp(position.x, -10, 10);
        rb.position = position;
    }
    private void Flip()
    {
        facingRight = !facingRight;
        transform.Rotate(0, 180, 0);
    }
}
```

Після цього перетягуємо скрипт на персонаж (Player).

Контрольні питання:

1. Що таке анімація?
2. Які кадри називаються ключовими (key frames)?
3. Що таке атлас зі спрайтами?
4. Назвіть етапи анімації персонажа.
5. Чим відрізняться спрайтова анімація від скелетної анімації?
6. Що таке сумісна анімація (швидкість, час і deltaTime)?
7. Що таке Accet Unity, де їх скачати і як додати до проекту?

Самостійна робота:

Кістякова 2D анімація

Кістякова анімація 2D - це метод анімації персонажів, який використовує систему кісток для керування рухом їхніх частин тіла. Цей метод може використовуватися для створення реалістичних та плавних анімацій 2D персонажів.

Основні кроки для створення кістякової 2D **анімації:**

1. **Створення персонажу:**

- Можна намалювати свій персонаж або використовувати готовий спрайт.

- Розбийте персонаж на частини тіла, такі як голова, тулуб, руки, ноги тощо.
- З'єднайте частини тіла за допомогою кісток.

2. **Створення анімації:**

- Для кожної анімації потрібно створити ключові пози.
- Ключові пози - це кадри, які визначають початковий і кінцевий стан анімації.
- Unity автоматично генерує проміжні кадри між ключовими позами.

3. **Налаштування анімації:**

- Можна налаштувати швидкість анімації, петлі анімації, тригери анімації тощо.
- Можна використовувати різні компоненти Unity, такі як Animator та Animation Controller, для керування анімаціями.

Ресурси, які допоможуть дізнатися більше про кістякову 2D анімацію:

- **Офіційна документація Unity:**

[https://docs.unity3d.com/ru/530/Manual/UsingHumanoidChars.html]

- **Уроки** Unity**:**

[https://blog.unity.com/engine-platform/getting-started-with-2d-animation-package; https://vionixstudio.com/2023/03/14/unity-2d-animation/; https://vionixstudio.com/2023/03/14/unity-2d-animation/]

Кістякова 2D анімація- це потужний інструмент, який може допомогти створювати реалістичні та плавні анімації 2D персонажів.

Декілька порад:

- Використовуйте якнайменше кісток, щоб зробити персонаж легким для анімації.
- Створіть чітку ієрархію кісток.
- Використовуйте інструменти Unity для автоматизації анімації.
- Тестуйте анімації та вносите необхідні зміни.