Міністерство освіти і науки України
Одеська національна академія харчових технологій
Кафедра комп’ютерної інженерії

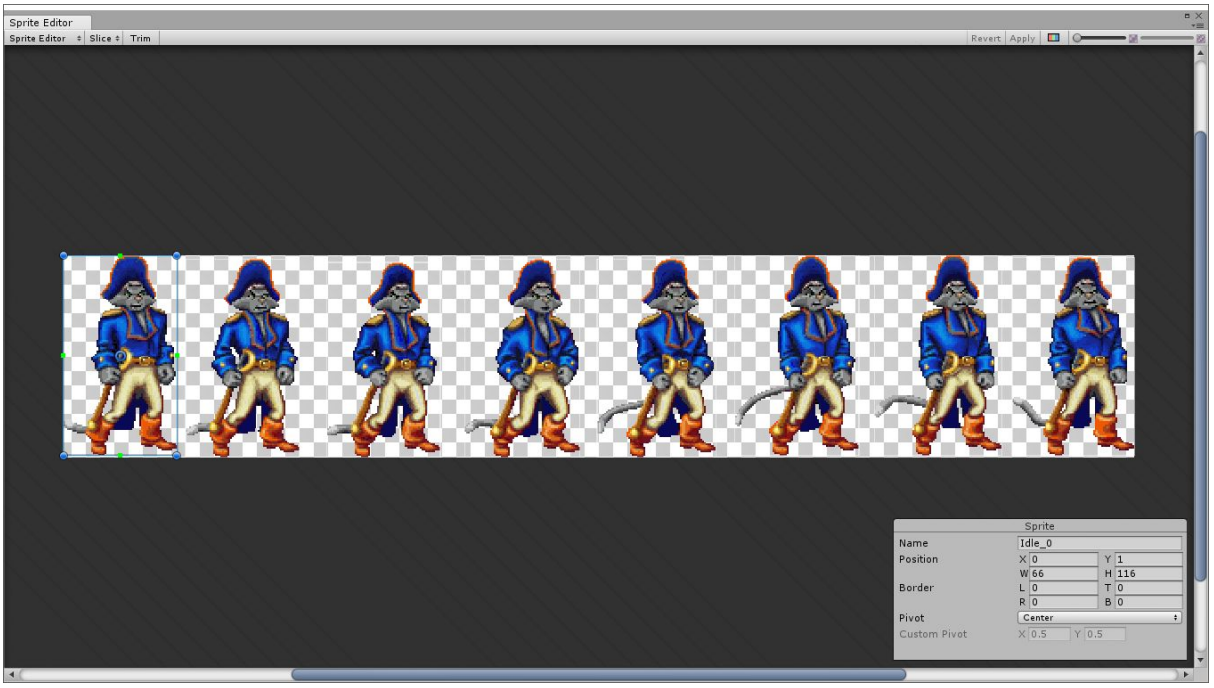
Лабораторна робота №2

з дисципліни «Проектування ігрових систем»

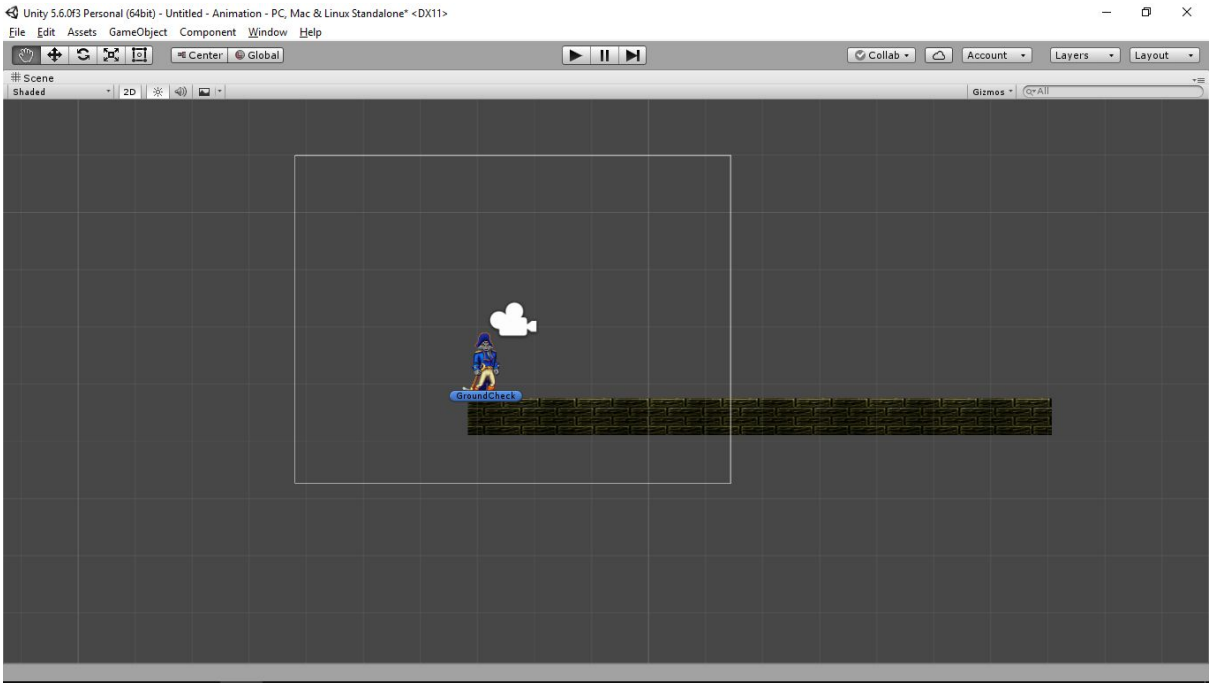
Виконав: студент групи 542
Погосяна Михайла.
Перевірив: Сіренко О.І.

Одеса 2017 р.

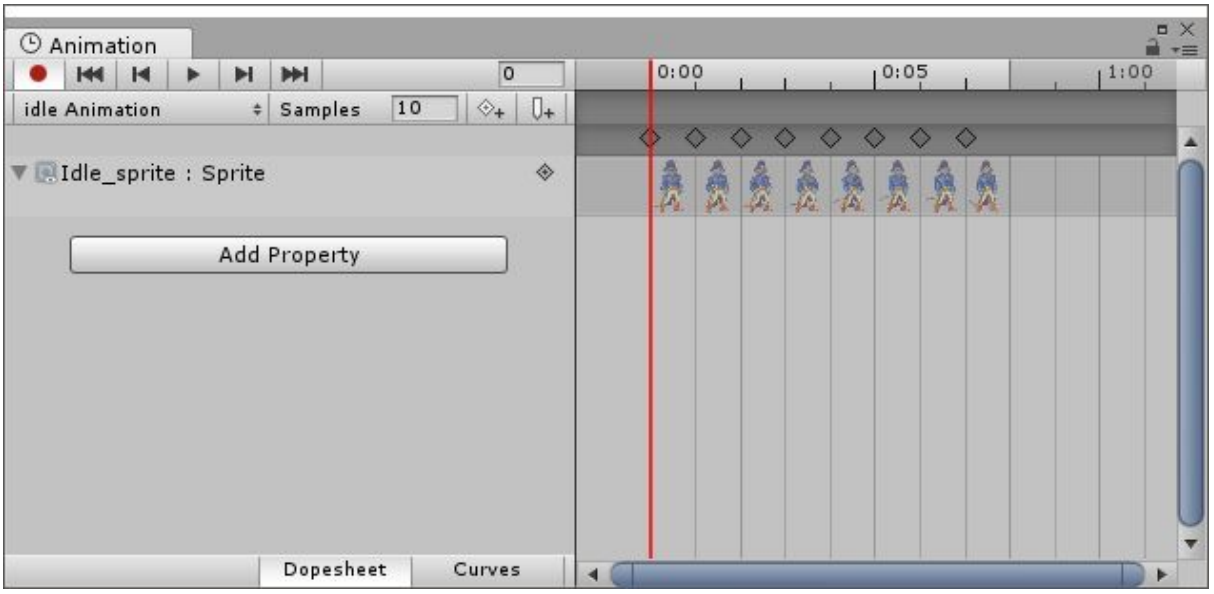
Створення анімації для Idle:
Нарізка спрайтшита для анімації в Sprite Editor.



Створюємо персонажа по першому кадру:

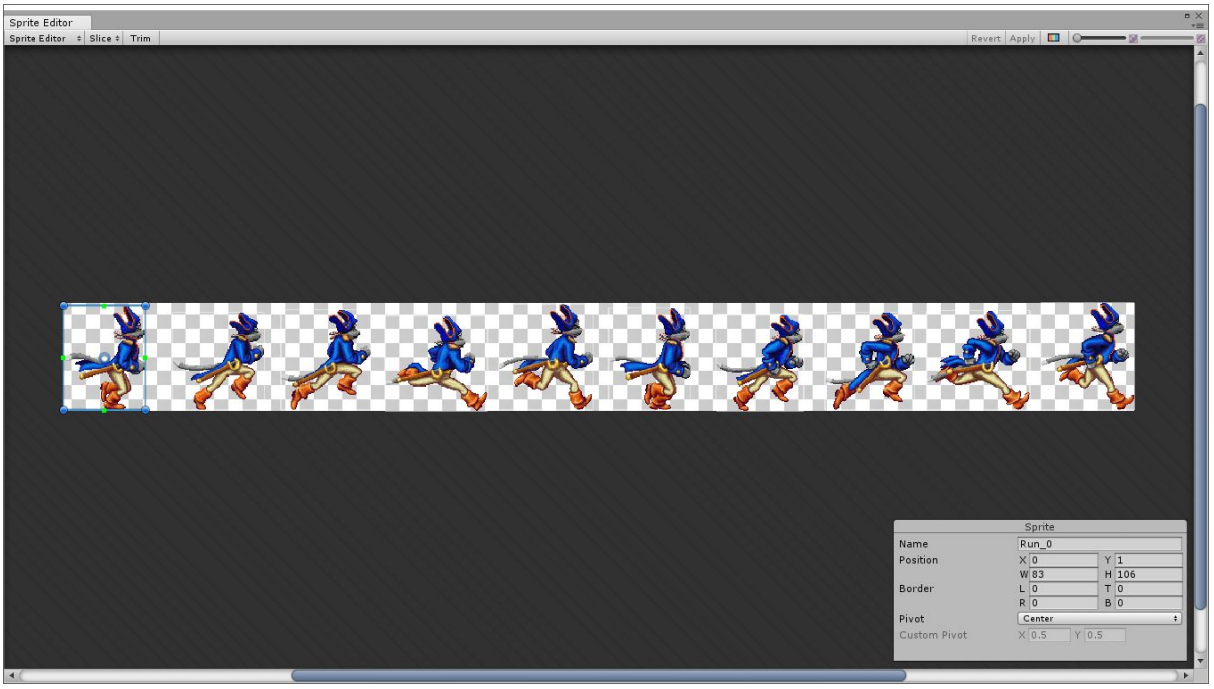


Анімація для idle режиму

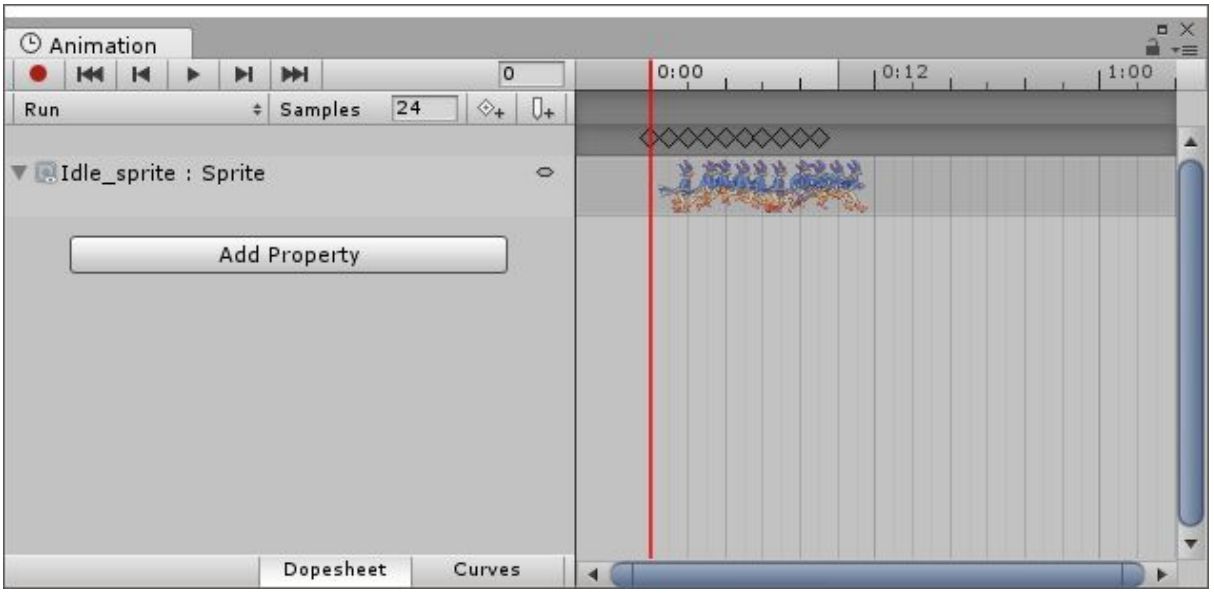


Створення анімації бігу:

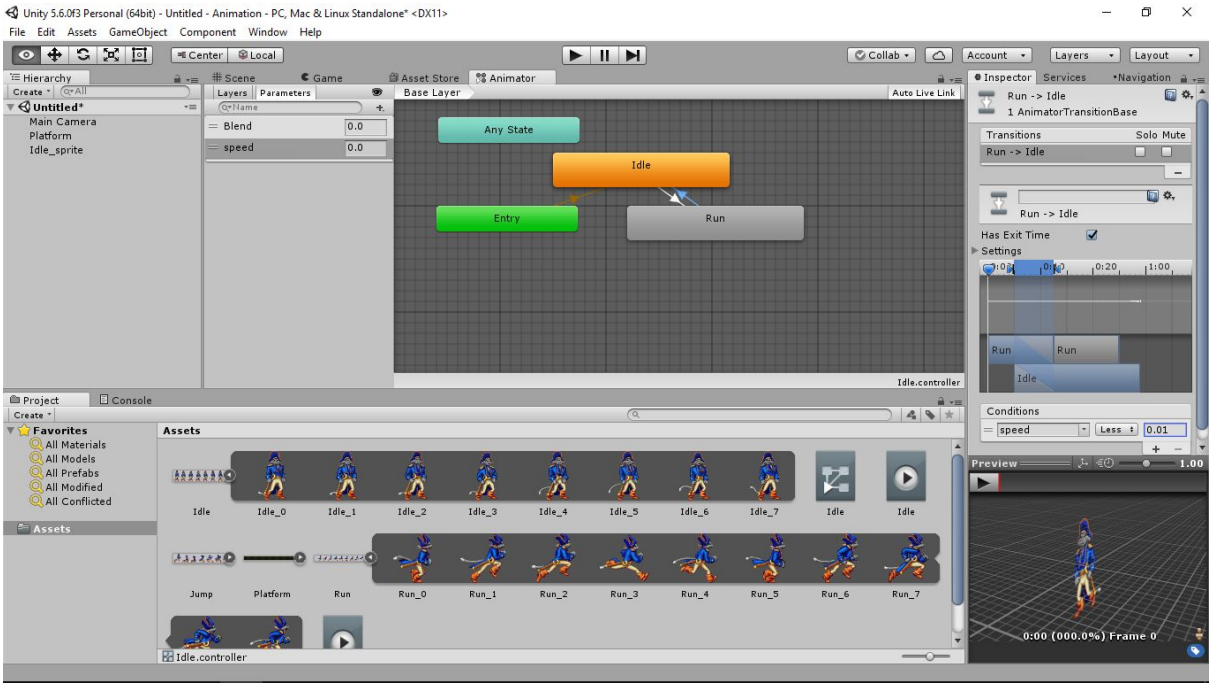
Створемо спрайтшит



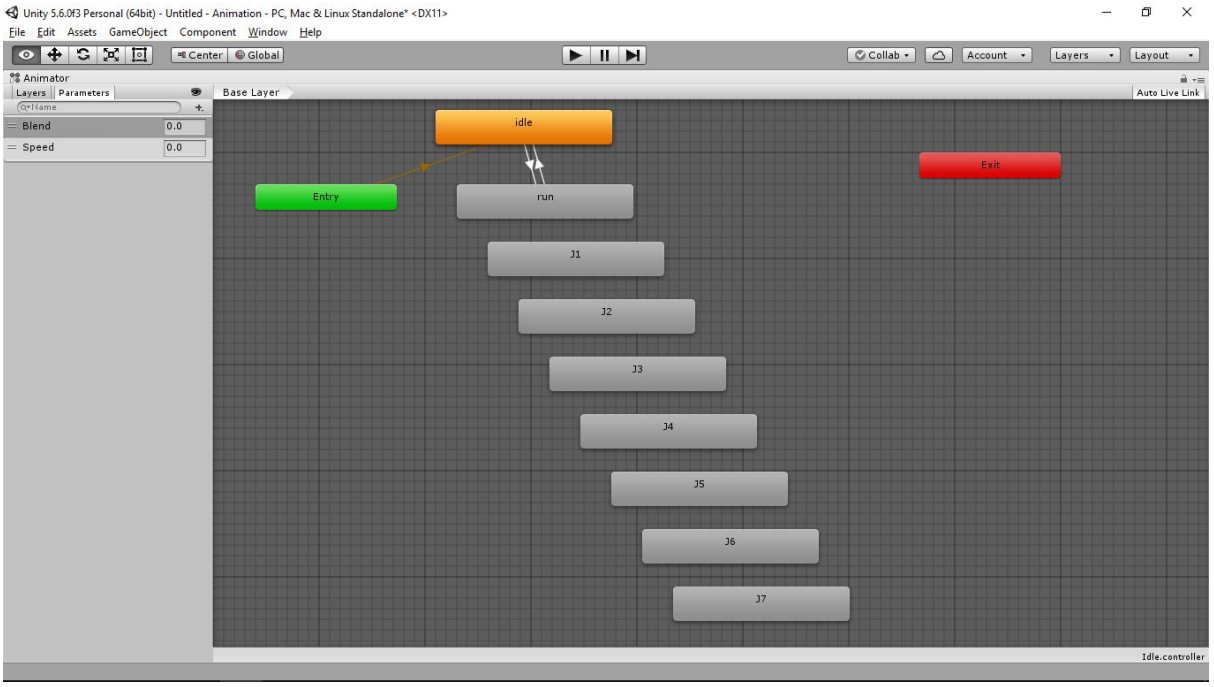
Створюємо анімацію бігу



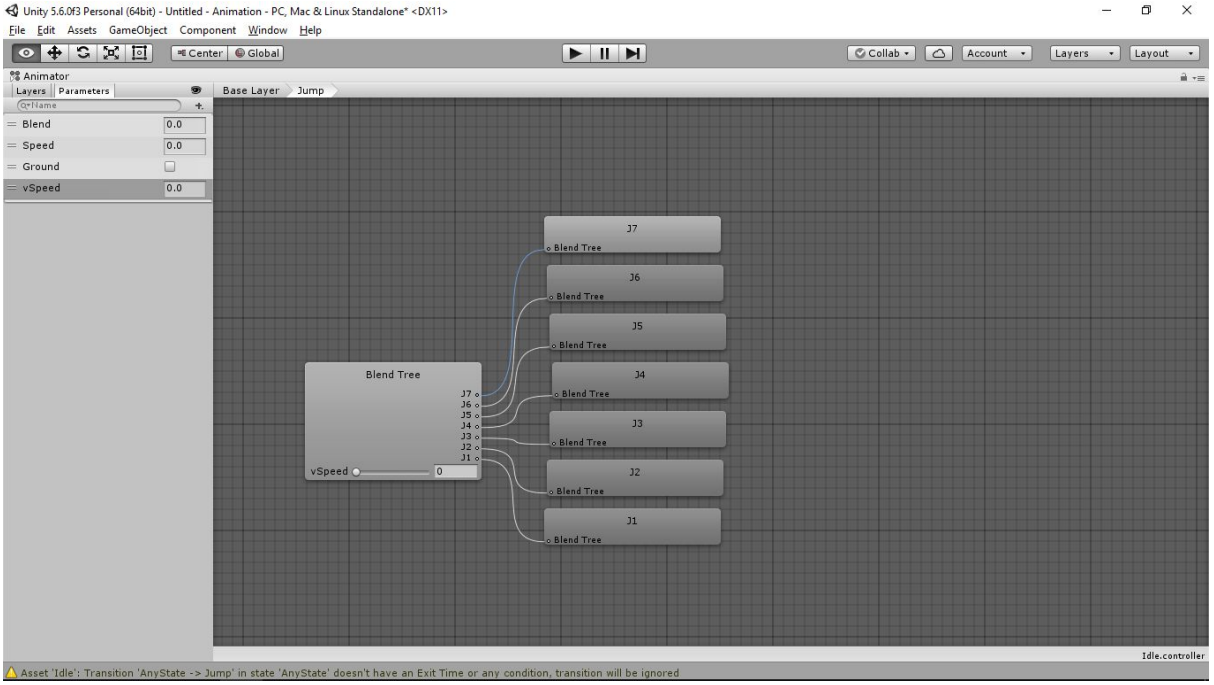
Встановлюємо параметри переходу



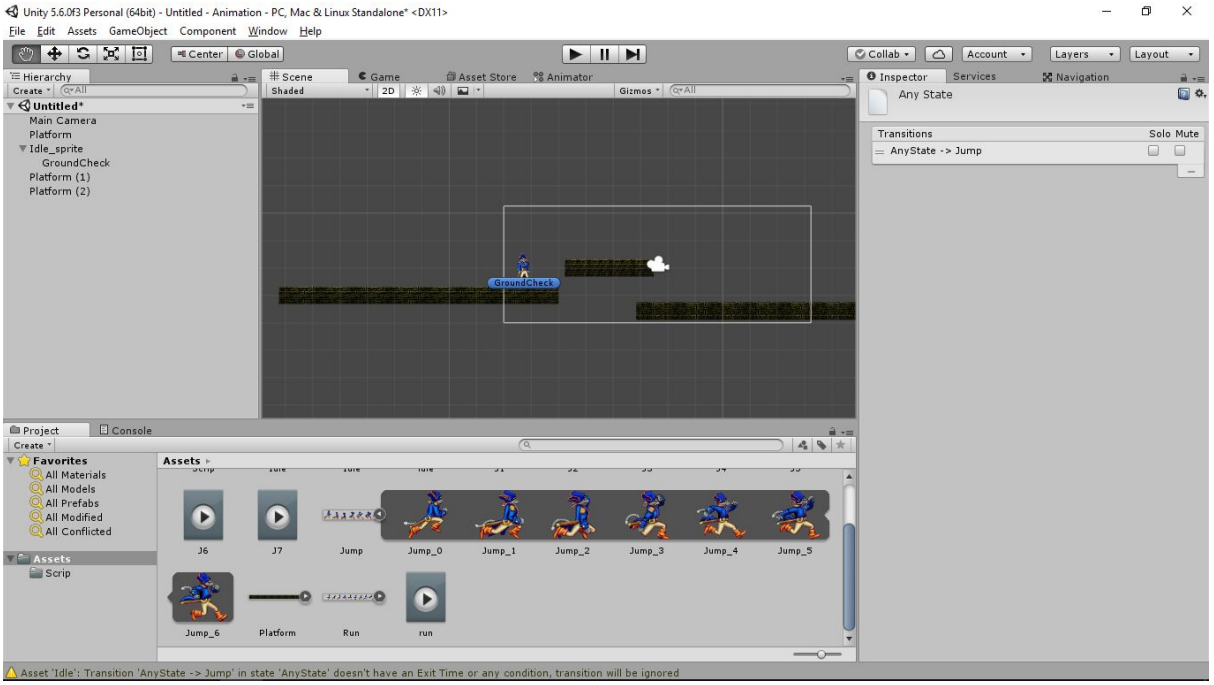
Реалізація стрибка та падіння:
Створимо 7 анімацій по 1 кадру



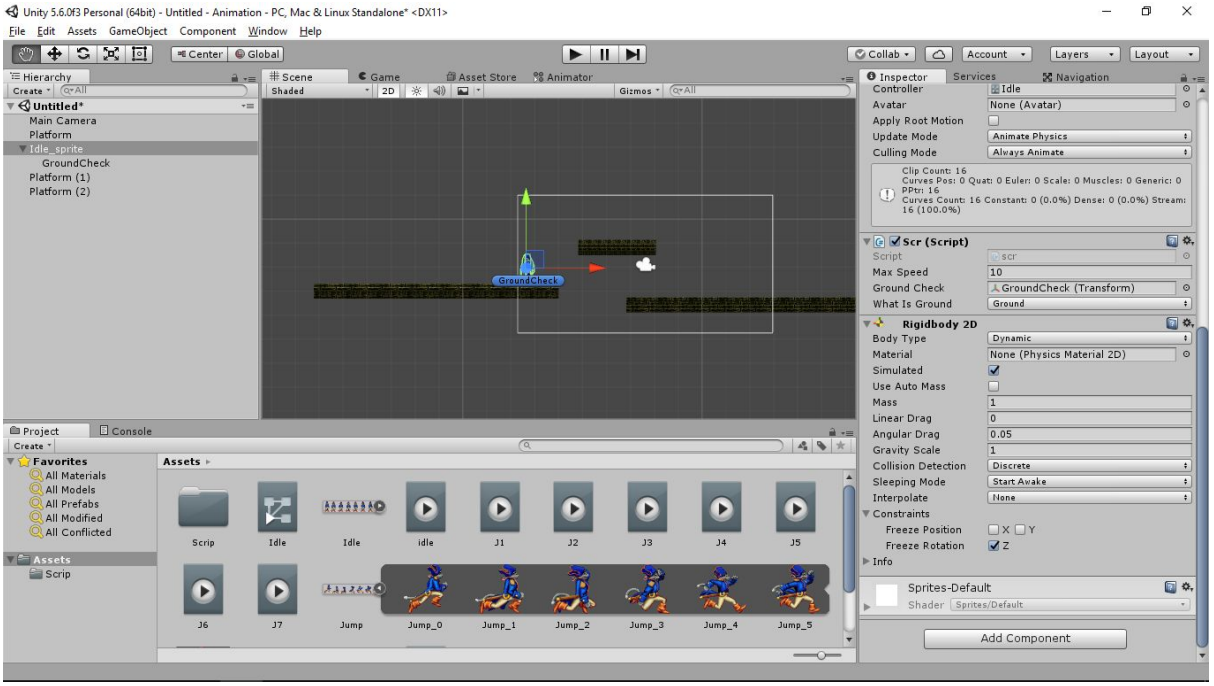
Реалізація Blend Tree в анімації jump



Добавимо платформ



Налаштування скрипта



Скрипт

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

public class scr : MonoBehaviour
{
    //переменная для установки макс. скорости персонажа
    public float maxSpeed = 10f;
    //переменная для определения направления персонажа вправо/влево
    private bool isFacingRight = true;
    //ссылка на компонент анимаций
    private Animator anim;
    //находится ли персонаж на земле или в прыжке?
    private bool isGrounded = false;
    //ссылка на компонент Transform объекта
    //для определения соприкосновения с землей
    public Transform groundCheck;
    //радиус определения соприкосновения с землей
    private float groundRadius = 0.2f;
    //ссылка на слой, представляющий землю
    public LayerMask whatIsGround;

    /// <summary>
    /// Начальная инициализация
    /// </summary>
    private void Start()
    {
        anim = GetComponent<Animator>();
    }

    /// <summary>
    /// Выполняем действия в методе FixedUpdate, т. к. в компоненте Animator
персонажа
    /// выставлено значение Animate Physics = true и анимация синхронизируется с
расчетами физики
    /// </summary>
    private void FixedUpdate()
    {//определяем, на земле ли персонаж
        isGrounded = Physics2D.OverlapCircle(groundCheck.position, groundRadius,
whatIsGround);
        //устанавливаем соответствующую переменную в аниматоре
        anim.SetBool ("Ground", isGrounded);
        //устанавливаем в аниматоре значение скорости взлета/падения
        anim.SetFloat ("vSpeed", GetComponent<Rigidbody2D>().velocity.y);
        //если персонаж в прыжке - выход из метода, чтобы не выполнялись действия,
связанные с бегом
        if (!isGrounded)
```

```
            return;

        //используем Input.GetAxis для оси X. метод возвращает значение оси в пределах от -1 до 1.
        //при стандартных настройках проекта
        //-1 возвращается при нажатии на клавиатуре стрелки влево (или клавиши A),
        //1 возвращается при нажатии на клавиатуре стрелки вправо (или клавиши D)
        float move = Input.GetAxis("Horizontal");

        //в компоненте анимаций изменяем значение параметра Speed на значение оси X.
        //приэтом нам нужен модуль значения
        anim.SetFloat("Speed", Mathf.Abs(move));

        //обращаемся к компоненту персонажа RigidBody2D. задаем ему скорость по оси X,
        //равную значению оси X умноженное на значение макс. скорости

        GetComponent<Rigidbody2D>().velocity = new Vector2(move * maxSpeed, GetComponent<Rigidbody2D>().velocity.y);

        //если нажали клавишу для перемещения вправо, а персонаж направлен влево
        if(move > 0 && !isFacingRight)
            //отражаем персонажа вправо
            Flip();
        //обратная ситуация. отражаем персонажа влево
        else if (move < 0 && isFacingRight)
            Flip();
    }
    private void Update()
    {
        //если персонаж на земле и нажат пробел...
        if (isGrounded && Input.GetKeyDown (KeyCode.Space))
        {
            //устанавливаем в аниматоре переменную в false
            anim.SetBool("Ground", false);
            //прикладываем силу вверх, чтобы персонаж подпрыгнул
            gameObject.GetComponent<Rigidbody2D> ().AddForce(new Vector2(0, 600));
        }
    }

    /// <summary>
    /// Метод для смены направления движения персонажа и его зеркального отражения
    /// </summary>
    private void Flip()
    {
        //меняем направление движения персонажа
        isFacingRight = !isFacingRight;
```

```
        //получаем размеры персонажа
        Vector3 theScale = transform.localScale;
        //зеркально отражаем персонажа по оси X
        theScale.x *= -1;
        //задаем новый размер персонажа, равный старому, но зеркально отраженный
        transform.localScale = theScale;
    }
}
```