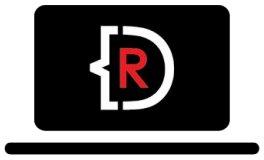
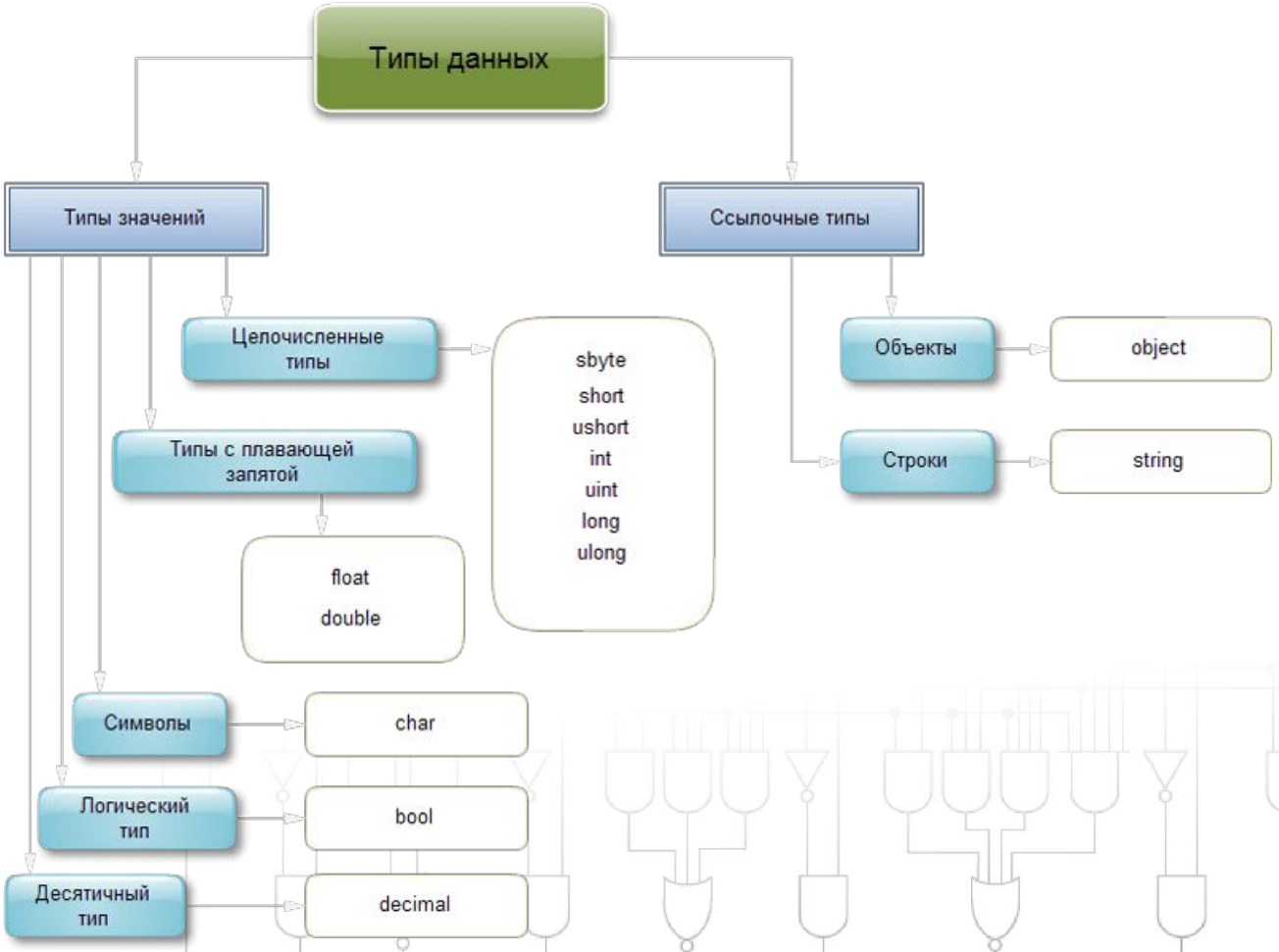
Game design



<{DeveloperR/>

Повторення

Які типи даних C# ви знаєте?



Повторення

В чому різниця між Update () та FixedUpdate()?

Update () фикликається що фрейму (ФПС, шви залежать від різних умов), тому інтервали виклику функції можуть відрізнятися. Виклик FixedUpdate відбувається через сталі інтервали часу, тому після кожного виклику функції відбуваються необхідні розрахунки фізики

Повторення

Для чого використовується структура Vector3(x, y, z)?

Ця структура використовується всередині Unity для передачі 3D-позицій та напрямків. Вона також містить функції для виконання загальних векторних операцій.

Повторення

***Для чого і в яких випадках використовується умова* else if?**

Це додаткова умова, що перевіряється, після того, як умова if не виконалась.

Генерація випадкових об’єктів

Для створення більш цікавої (непередбачуваної) динаміки гри дуже часто використовується генерація випадкових чисел (рандомізація).

Це може бути:

Генерація ворогів

Генерація значень урону

Генерація випадкових об’єктів

Скрипт випадкової генераціїї об’єктів додають до незнищуваного порожнього об’єкта, камери, фону, тощо, вказуючи префаб об’єкта який треба генерувати.

Оголошуємо змінні:

```
public GameObject ball; //об’єкт, який будемо генерувати

float offsetX = 0; //початкове положення по X

float offsetY = 10f; // початкове положення по Y

float fTime = 1f; //частота створення «клонів»
```

Генерація випадкових об’єктів

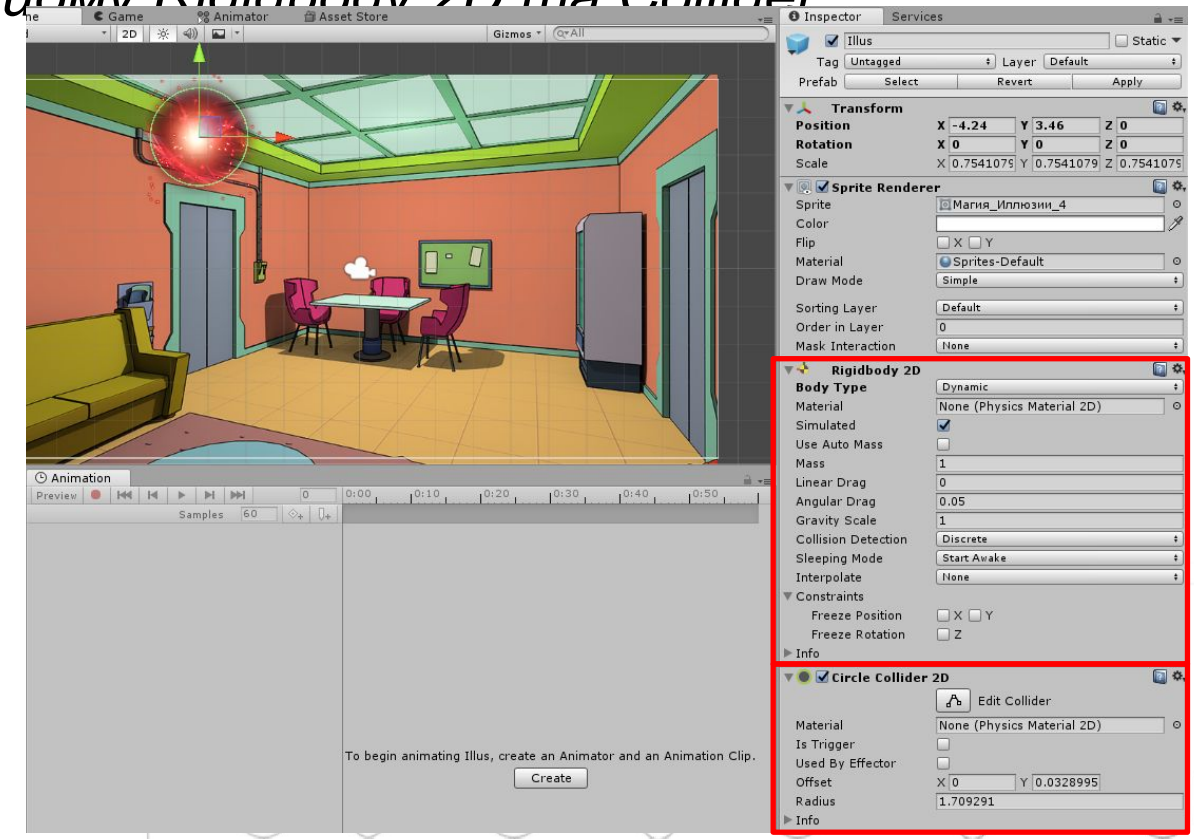
У певному проміжку часу задаємо випадкову генерацію об’єкту в діапазоні значень по осі X.

```
void Update () {
    if (fTime < Time.realtimeSinceStartup)
    {
      float offsetX = Random.Range(-10f,10f);
      Instantiate(baloon, new Vector3(offsetX, 10f, -5),
Quaternion.identity);
      fTime = Time.realtimeSinceStartup + Random.Range(3f, 5f);
    }
}
```

Границі генерації випадкових значень

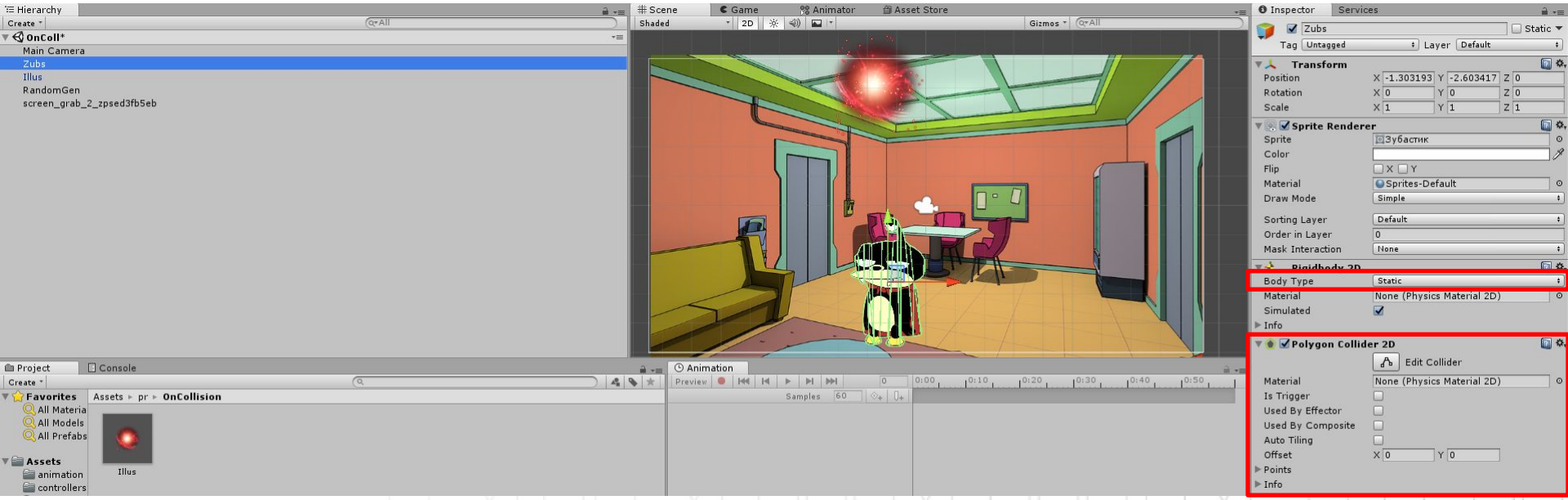
Завдання

- *Створіть фон екрану (jpg картинка).*
- *Створіть пороожній GameObject, що падатиме.*
- *Задайте йому Rigidbody 2D та Collider*



Завдання

- *Створіть Головного героя (ігровий персонаж).*
- *Задайте йому також Rigidbody (тип – Static) та колаедр.*
- *Створіть скрипт керування об’єктом.*



Знищення об’єктів

Напишемо скрипт, що детектуватиме контакт об’єктів і знищуватиме один з них.

Якщо обидва об’єкти мають колаедри – то найпростіший спосіб це зробити – використати функцію OnCollisionEnter.

```
void OnCollisionEnter2D(Collision2D col)
{
    /*Тіло функції*/
}
```

Знищення об’єктів

При контакті двох колаедрів задамо перевірку, що дозволить обирати колаедри лише з певним ім’ям і знищувати лише ці об’єкти:

```
void OnCollisionEnter2D(Collision2D col)
    {
        if (col.gameObject.name == "Illus")
        {
            Destroy(col.gameObject);
        }
    }
```

Створюємо об’єкт класу Collision2D

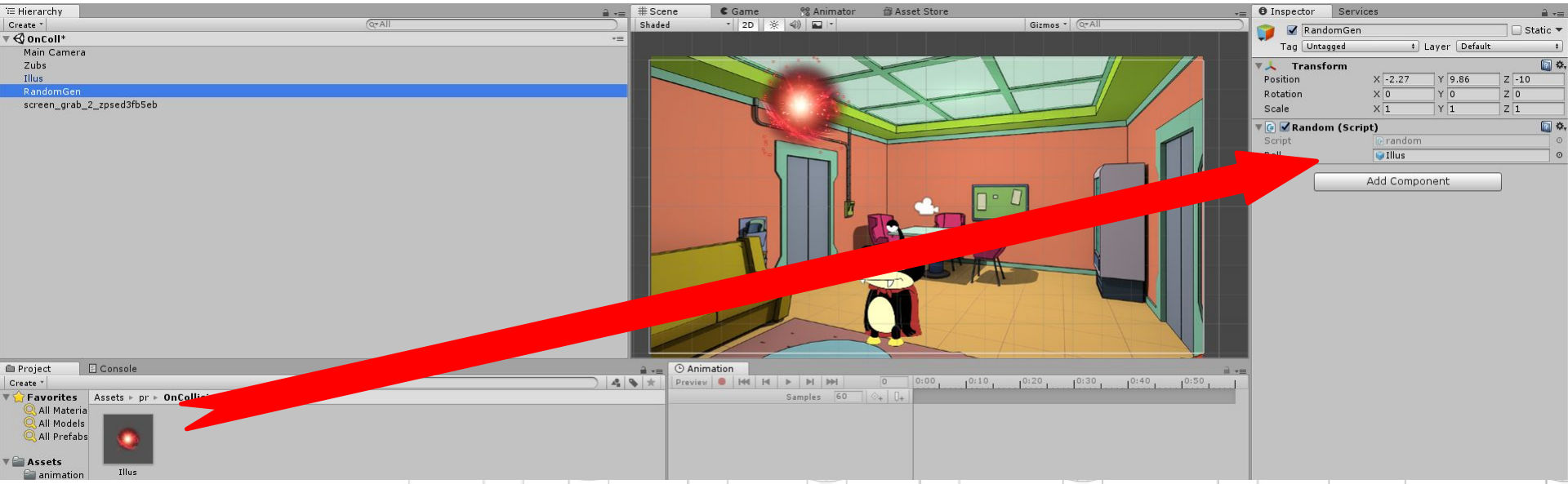
Перевірка взаємодії з колаедром об’єкта з ім’ям “Illus”

Якщо ім’я співпало - знищуємо об’єкт, з яким взаємодіяли

Запустіть гру та перевірте

Завдання

- *Напишіть скрипт для випадкової генерації тіл.*
- *Підв’яжіть скрипт до порожнього GameObject.*
- *Задайте префаб об’єкту, що буде генеруватись.*
- *Проаналізуйте діапазони та частоту створення випадкових*



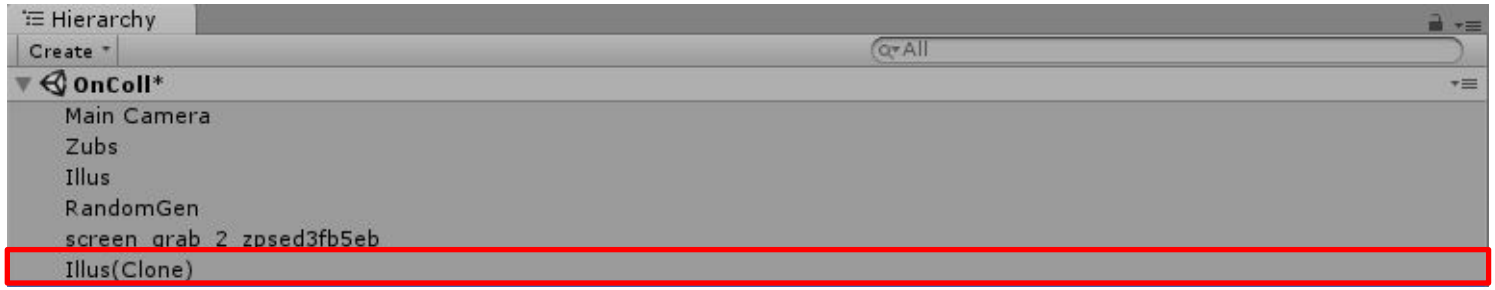
Завдання

```
public GameObject ball;
float offsetX = 0;
float offsetY = 10f;
float fTime = 1f;

void Start()
    {

    }
void FixedUpdate()
    {
        if (fTime < Time.realtimeSinceStartup)
        {
            offsetX = Random.Range(-8f, 9f);
            Instantiate(ball, new Vector3(offsetX, offsetY, 0),
Quaternion.identity);
            fTime = Time.realtimeSinceStartup + Random.Range(2f, 3f);
        }
    }
```

Завдання

Запустіть гру та перевірте, як працює і чи знищуються згенеровані об’єкти при контакті?

Знищується лише перший об’єкт, що додали на сцену, сгенеровані об’єкти контактують але не знищуються. Це пов’язано з особливостями генерації:



При генерації об’єктів до імені додається приписка (Clone), слід врахувати в скрипт.

Завдання

Змінимо функцію `OnCollisionEnter2D`:

```
void OnCollisionEnter2D(Collision2D col)
    {
        if (col.gameObject.name == "Illus(Clone)")
        {
       Destroy(col.gameObject);
        }
    }
```

Тепер відбувається знищення всіх об'єктів крім першого, його можна просто видалити.

Знищення об’єктів

Метод детектування детектування взаємодії `OnCollisionEnter` має ряд недоліків:

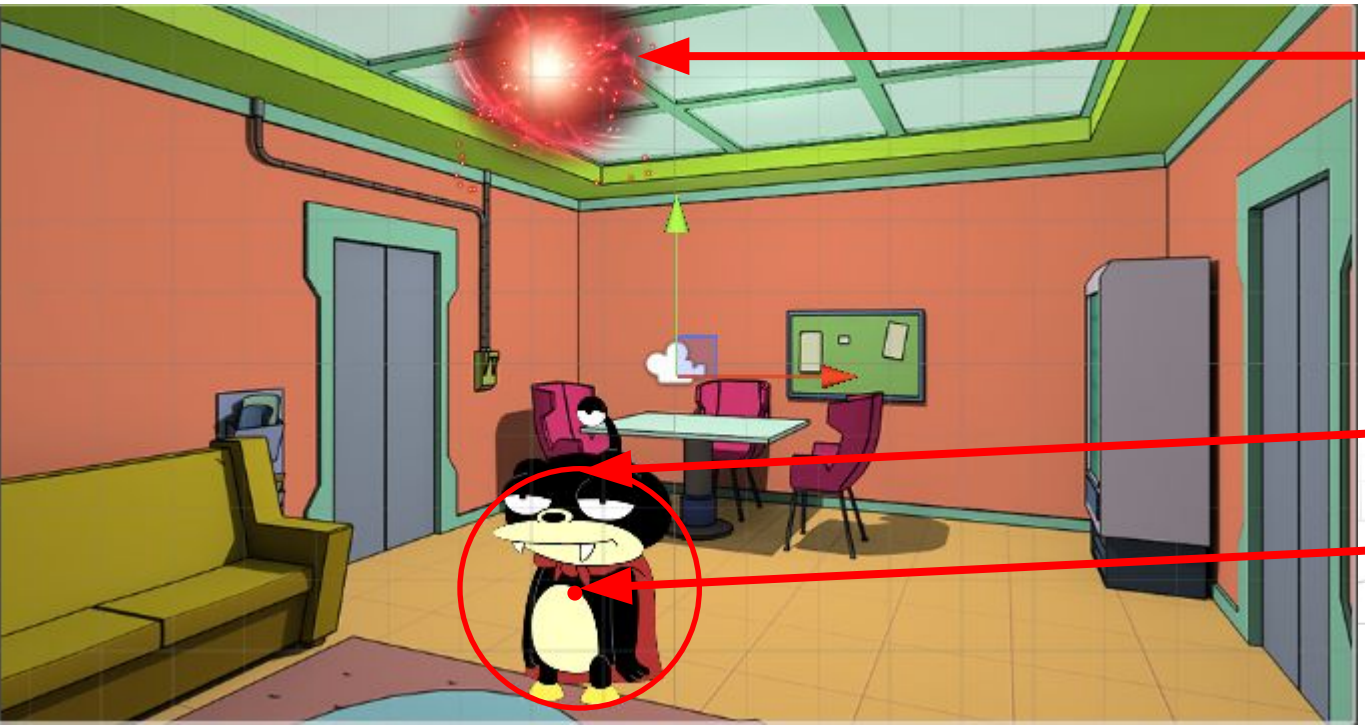
- *один з об’єктів, що взаємодіють обов’язково повинен мати не kinematic/static rigidbody.*
- *будь-який інший об’єкт, у якого є колаедр відчуватиме вплив заданого.*

Знищення об’єктів

Вище наведені пункти значно ускладнюють процес реалізації детектування взаємодій, особливо, якщо йдеться про постріли та Friendly fire. А в деяких іграх персонажам взагалі не задають колаедри, щоб позбутися можливих «блокувань».

Знищення об’єктів

Для реалізації вище наведених прикладів краще використати детектування так званних LayerMask. Для цього слід провести перевірку входження детектуємого шару об’єкта в окіл уявної точки, що підв’язана до основного ігрового обє’кта:



Знищення об’єктів

Створимо public змінні, в яких зможемо задати прив’язку уявної точки до об’єкта, радіус перевірки, та шар, який будемо детектувати:

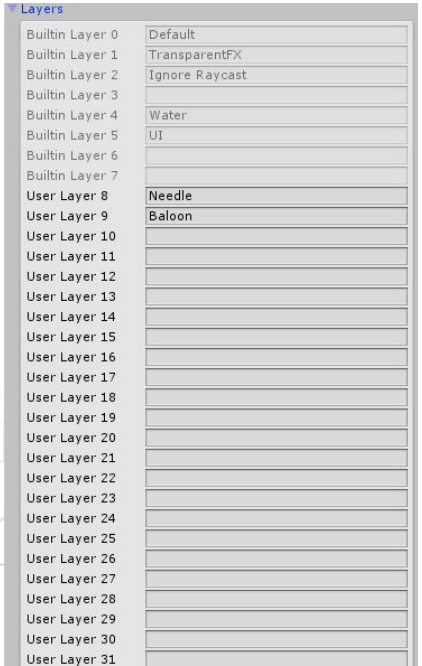
```
    public Transform zubiCheckPoint; // точка перевірки контакту з об’єктом
    public float zubiCheckRadius; // радіус перевірки контакту з об’єктом
    public LayerMask zubiCheckLayer; // прошарок
    private bool isTouchingzubi; // змінна для зберігання результату перевірки
```

Знищення об’єктів

LayerMask - *можуть бути використані для вибіркового фільтрування об'єктів гри, наприклад, при проходженні променів:*

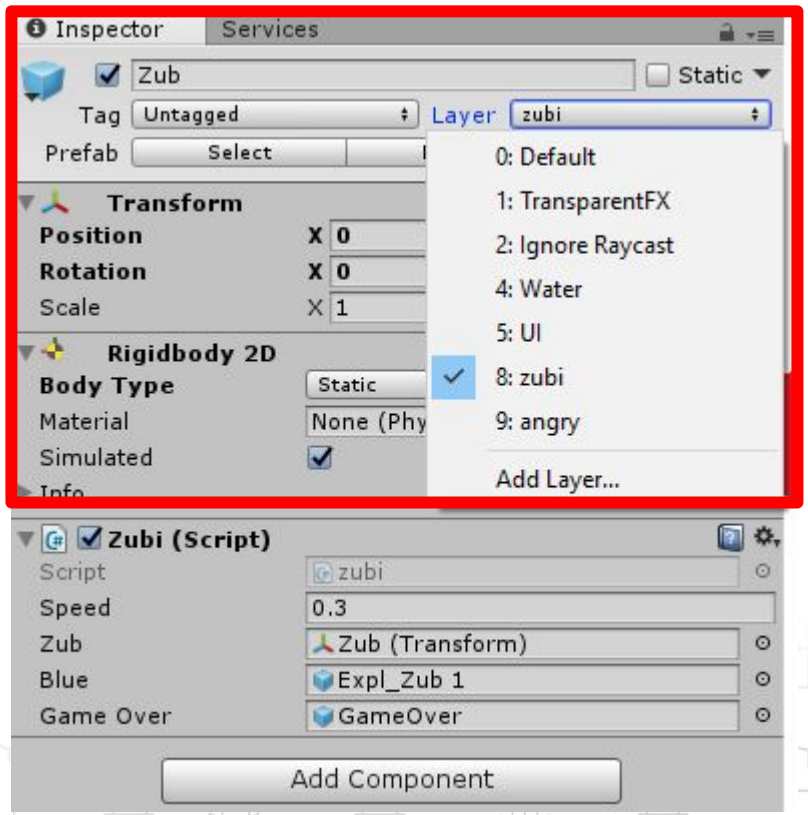
- *постріли (friendly fire);*
- *Взаємодії об’єктів в обхід колаєдра, тощо.*

Всього налічується 32 шари (фактично 32 біти: активний та неактивний стан). Якщо в скрипті шар в активному стані – то він детектується.



Знищення об’єктів

Задайте персонажу новий шар:



Знищення об’єктів

Щоб визначити чи взаємодіють об’єкти. Слід визначити потрапляння об’єкту, що знаходиться на заданому шарі в уявне «коло» (OverlapCircle).

Коло визначається координатою його центру в загальних (світових) координатах та його радіусом. Додатковий параметр layerMask дозволяє перевіряти лише об'єкти на певних шарах.

```
isTouchingzubi =
Physics2D.OverlapCircle(zubiCheckPoint.position,
zubiCheckRadius, zubiCheckLayer);
```

Знищення об’єктів

Задамо умову знищення об’єкта при взаємодії з іншим (isTouchingzubi == true).

Також задамо умову знищення об’єкта, що вийшли за межі екрану та не були знищені основним ігровим об’єктом, щоб запобігти накопиченню «клонів» об’єкта:

```
if (isTouchingzubi)
    {
        Object.Destroy(this.gameObject);
    }
else if (transform.position.y < -7f)
    {
        Object.Destroy(this.gameObject);
    }
else { }
```

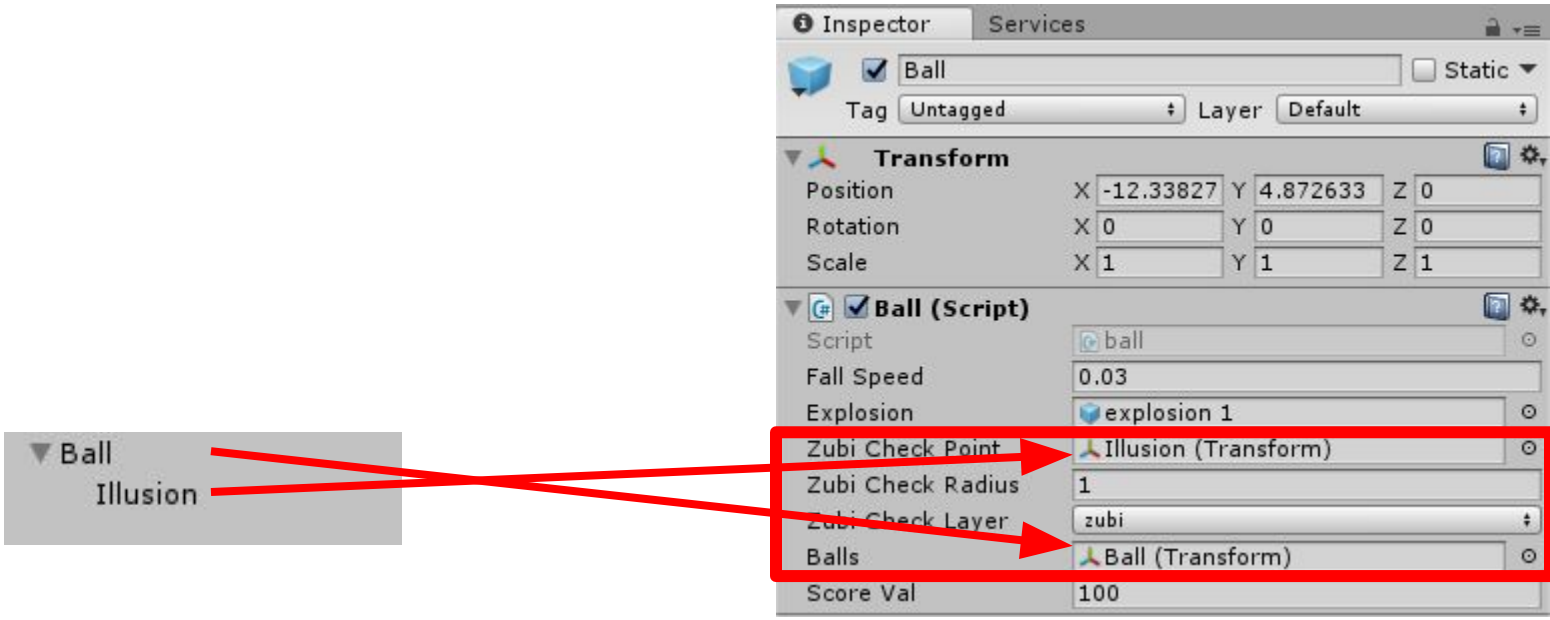
Завдання

- *Створіть порожній об’єкт і приєднайте до нього префаб кульки, що падатиме (кулька буде дочірньою до об’єкта).*

- *В скрипті, що відповідає за рух GameObject, що падатиме (у нас це Ball, скрипт можна взяти із попереднього уроку) додайте перевірку контакту з ігровим об’єктом.*
- *Додайте умови знищення GameObject при контакті із ігровим об’єктом («Жуйка»/Zubi) та при вичерпанні часу існування клону.*
- *Перетягніть скрипт на порожній об’єкт(Ball)*

Завдання

В Unity налаштуйте всі необхідні змінні:



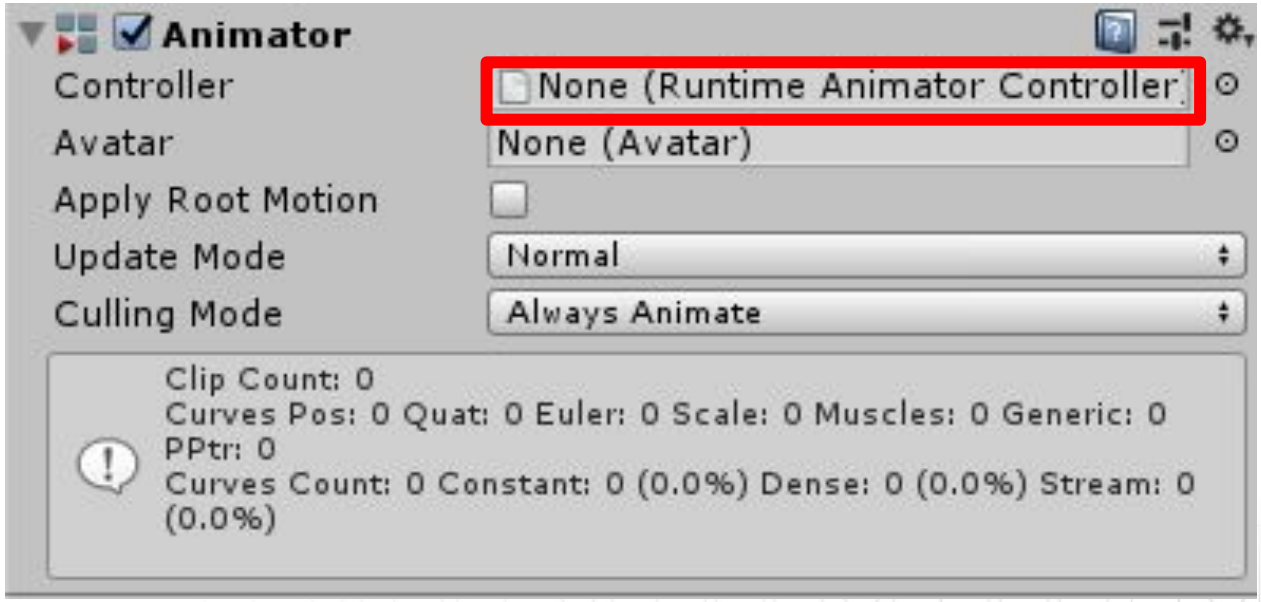
В якості шару, контакт з яким детектується – оберемо шар головного персонажу (zubi).

Завдання

Запустіть та перевірте

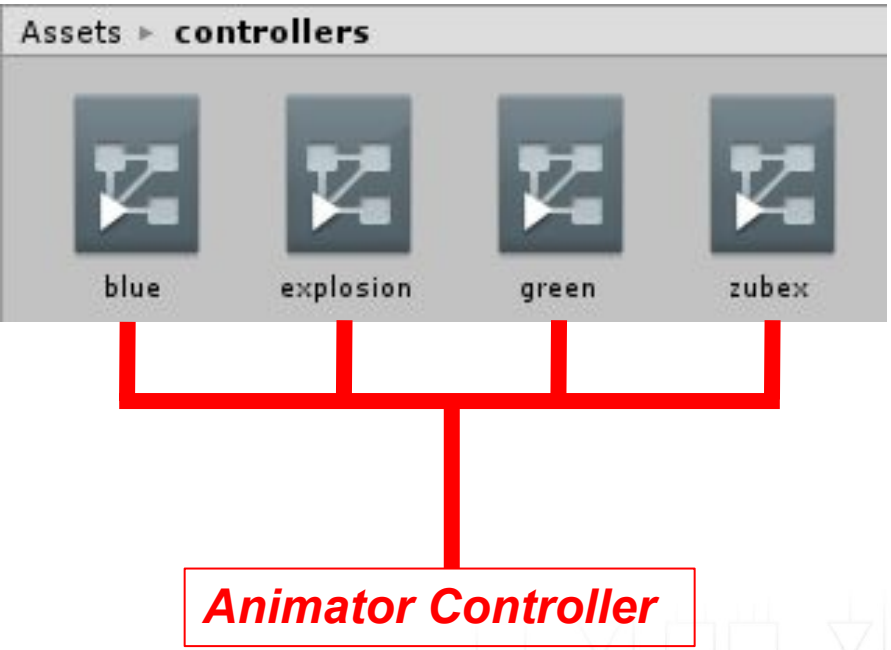
Анімація

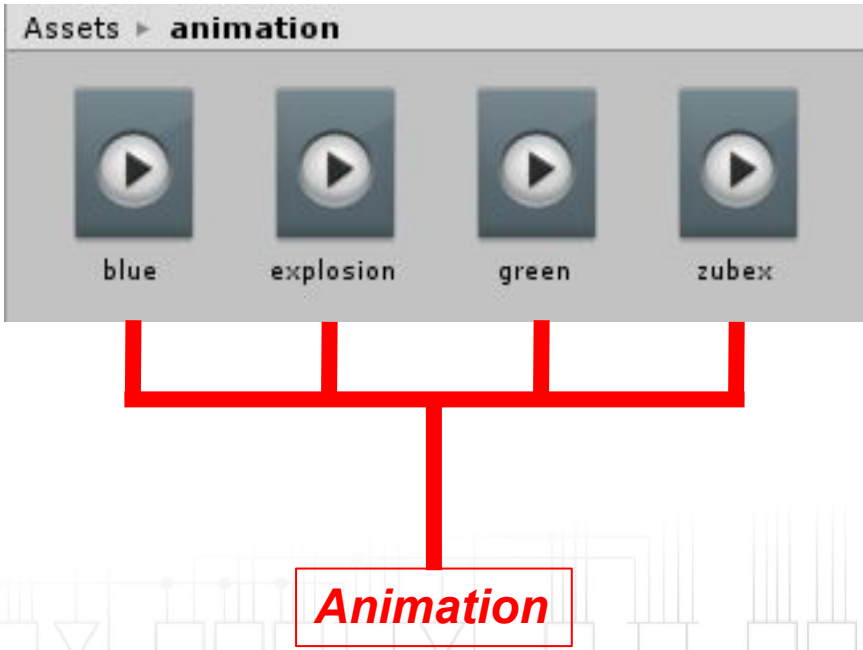
Створюємо новий порожній GameObject. Додаємо до нього компонент Animator (Miscellaneous)



Анімація

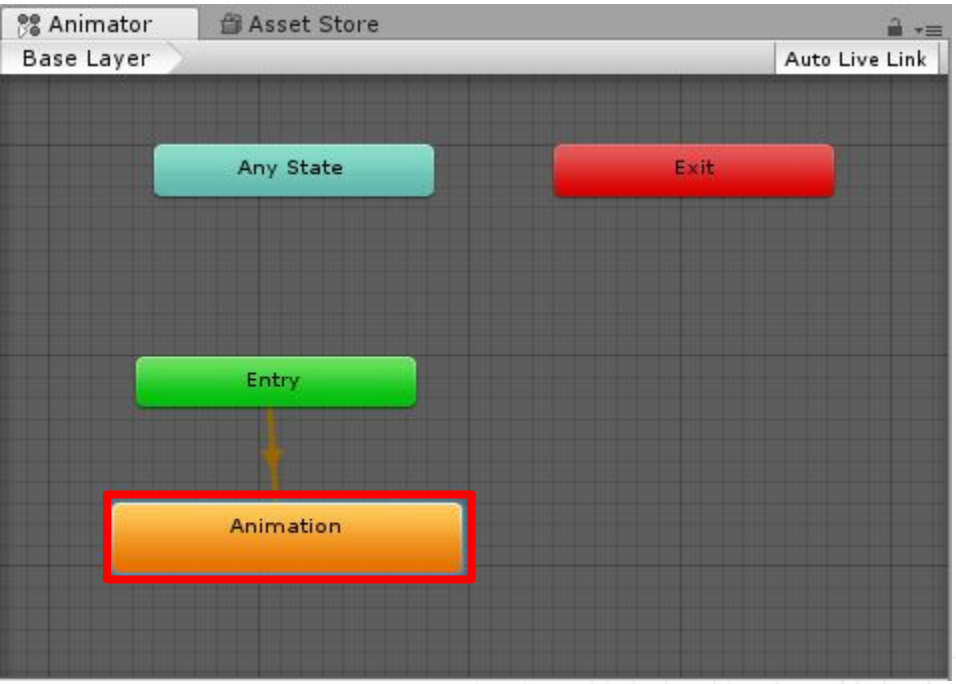
У папці Assets створюємо дві папки Controller та Animation в яких будуть зберігатись файли Animator Controller та Animation відповідно

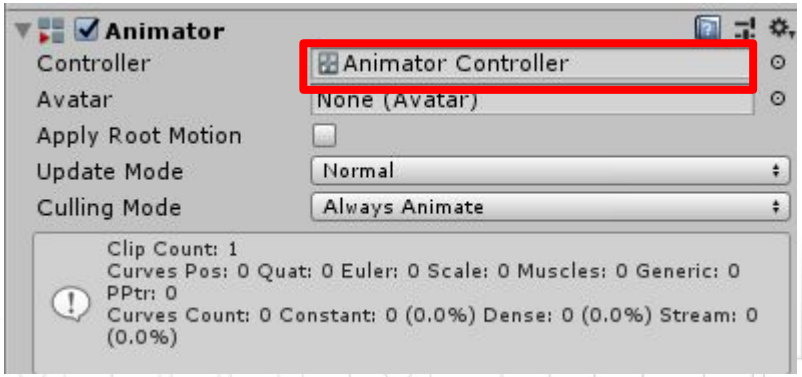




Анімація

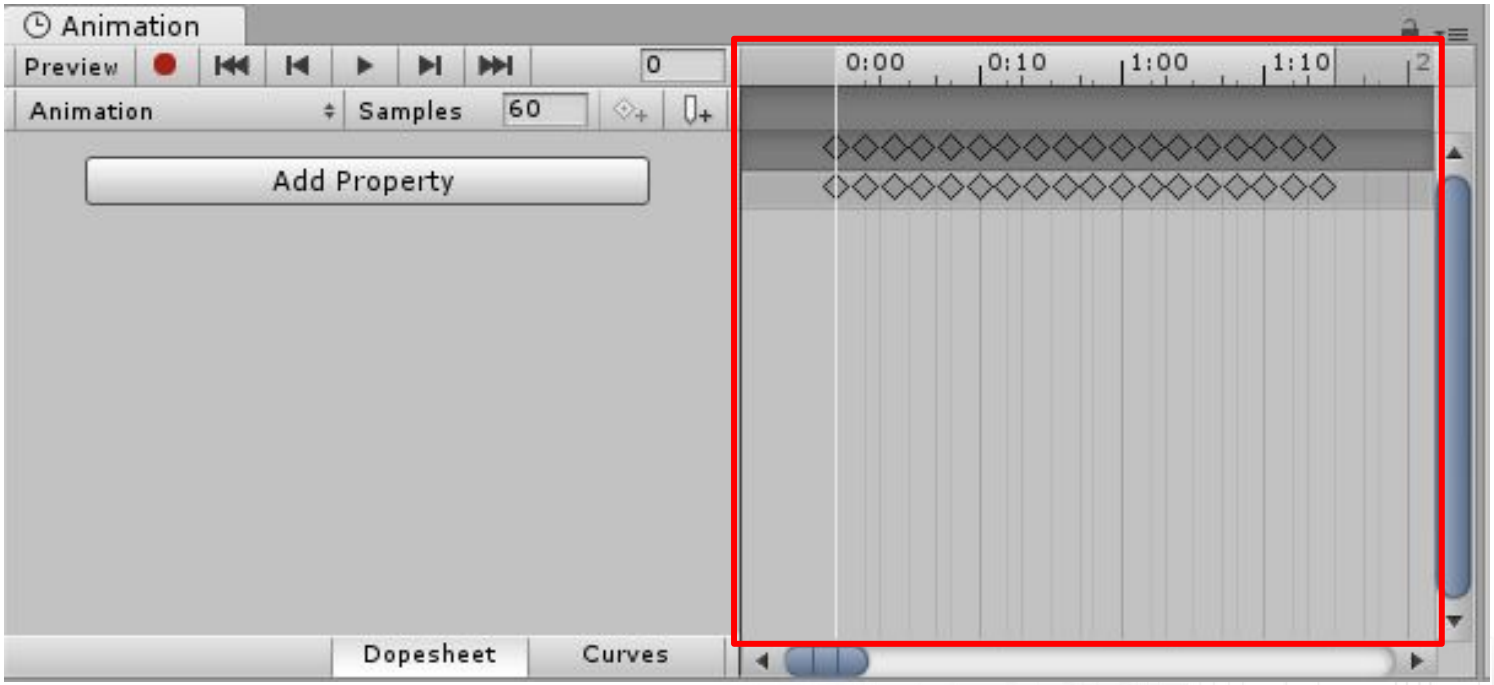
В файл Animator Controller додаємо створений раніше Animation. Сам файл Animator Controller вставляємо у компонент Animator нашого порожнього GameObject





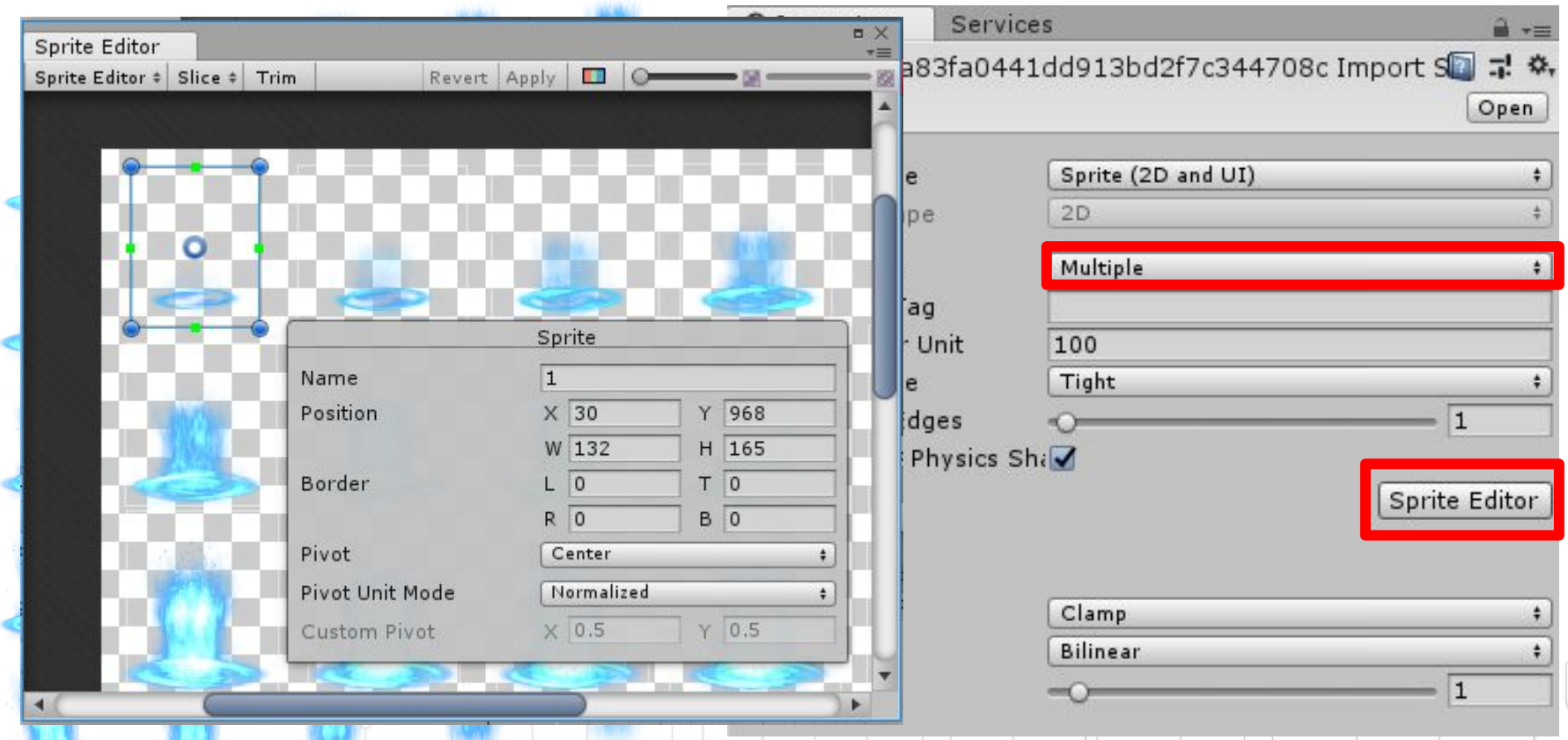
Анімація

Після виконаних дій вікно анімації стало активним і можна заповнити його готовими спрайтами.



Анімація

Якщо не має готових(порізаних покадрово) спрайтів, можемо створити вносячи зміни в заготовку.



Анімація

Зберігаємо префаб.

Підв'язуємо анімацію до об'єкту в якому вона повинна з'являтись.

Оголошуємо змінні:

```
public GameObject blue;
```

Анімація

Змінні для визначення поточного положення об'єкту до якого прив'язуємо анімації

```
float fcoordX;
float fcoordY;
```

```
fcoordX = zub.position.x;
fcoordY = zub.position.y;
```

Визначення положення об'єкта

```
 Instantiate(blue,new Vector3(fcoordX,fcoordY,0),
Quaternion.identity);
```

Функція виклику анімації на поточному місці об'єкта.

Завдання

- *Створи анімацію вибуху об’єкта*
- *У скрипт GameObject додай умову, за якою при знищенні об’єкту буде викликатися анімація вибуху.*
- *Самостійно додай анімації до гравця, в залежності від напряму руху гравця у сторони.*