ДЕРЖАНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни «Розробка комп'ютерних ігор»
Частина 1.

УДК **План НМВ 2019/2020рр.**

Рецензент -
Укладач –О.А. Дібрівний

ЗМІСТ

Лабораторна робота №1
Скриптування простих внутрішньо ігрових взаємодій «2D рух»

1. Постановка задачі

Створити просту ігрову сцену в 2D, та реалізувати рух об’єкту: вліво, вправо, стрибок, рух навприсідки.

2. Мета роботи

1. Розуміння поняття об’єкту на ігровій сцені та ігрових компонент.
2. Знайомство з компонентами Transform, Script Renderer, Rigidbody, Collider.
3. Отримання навичок з обробки відгуку від клавіатури та внесення змін до стану ігрових об’єктів на сцені.

3. Ключові положення

2.1 Знайомство з редактором Unity

На рис 1. Представлено головне вікно редактору Unity версії 2019.2.5f1. Для комфортної роботи з двигуном рекомендується встановлення наступних версій:

1. Версії 2019 року не раніше 2.5f1
2. Версія 2018 року не раніше 4.11f1
3. Версія 2017 року не раніше 4.33f1

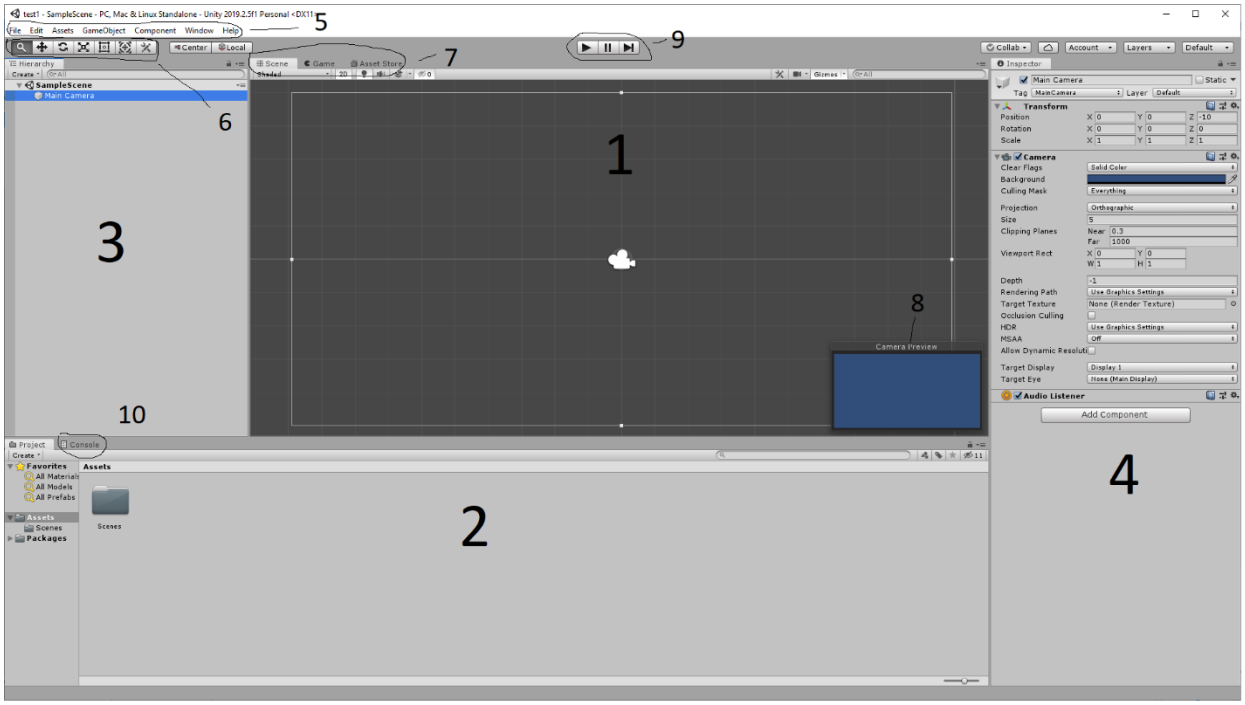

Рис.1. Головне вікно редактору Unity

Розглянемо окремі елементи цього вікна.

1 – Ігрова сцена, в середині якої розробник розміщує елементи для відображення та маніпуляцій в майбутній грі. Основними структурними елементами будь-якої 2D гри є спрайти(зображення). Unity підтримує зображення наступних форматів: PSD, TIFF, JPG, TGA, PNG, GIF, BMP, IFF, PICT, з максимальним розміром в текстури в пікселях 16384.

2 – Файловий провідник для папки з проектом Unity. Тут відображатимуться всі підгружені в проект файли такі як наприклад спрайти, скрипти та ігрові сцени.

3 – Вікно ієрархії. В цьому вікні відображатимуться всі елементи на конкретній ігровій сцені. По замовчуванню кожна ігрова сцена створюється з елементом Main Camera(головна камера). Цей об’єкт є точкою входу пристрою на ігрову сцену(по аналогії метод Main(); в C#).

4 – Вікно інспектору. В даному вікні відображаються всі компоненти, що підключені до конкретного ігрового об’єкту. Саме через підключення та налаштування компонент відбувається побудова взаємодій на ігровій сцені.

5 – Панель системних інструментів. Там знаходяться копки збереження, білду сцени, а також перехід до додаткових конфігураційних вікон.

6 – Панель режиму навігації по ігровій сцені.

7 – Панель преходу до вікон Game – погляд на ігрову сцену очима гравця та Unity Asset Store – офіційного ресурсу Unity з тисячами асетів(готових ігрових рішень, спрайтів, 3d моделей).

8 – Вікно погляду на ігрову сцену очима гравця(активне лише при виборі Main camera) в панелі ієрархії.

9 – Панель запуску гри, для тестування в режимі розробки.

10 – Перехід до вікна консолі(повідомлення про стан гри).

2.2 Поняття об’єкту на ігровій сцені та компонент

Об’єктом на ігровій сцені можна вважати будь-що що відображається в вікні ієрархії за винятком дочірніх об’єктів системного об’єкту Canvas які відносяться до UI(user interface). В будь-якого об’єкту на ігровій сцені по замовчуванню буде прикріплена компонента Transform(положення об’єкту в декартовому ігровому просторі). Решта компонент є опційними, як наприклад компонента Camera об’єкту Main Camera, що по суті являє собі фабричний скрипт для роботи з камерою.

2.3. Знайомство з компонентами Transform, Script Renderer, Rigidbody, Collider.

Компонента Transform(рис 2.) відповідає за положення ігрового об’єкту в межах декартового простору Unity.

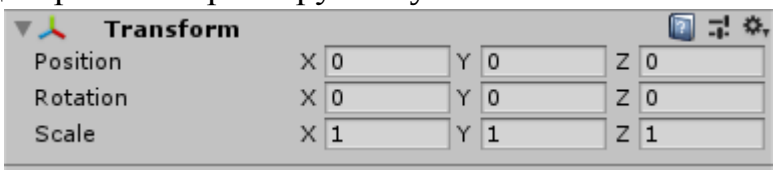

Рис.2 Вікно компоненти Transform.

.

Компонента Script Renderer(рис.3) дозволяє відображати зображення у вигляді спрайтів(Sprites), для використання на ігрових сценах. Дана компонента прикріплюється автоматично при додаванні будь-якого зображення на ігрову сцену.

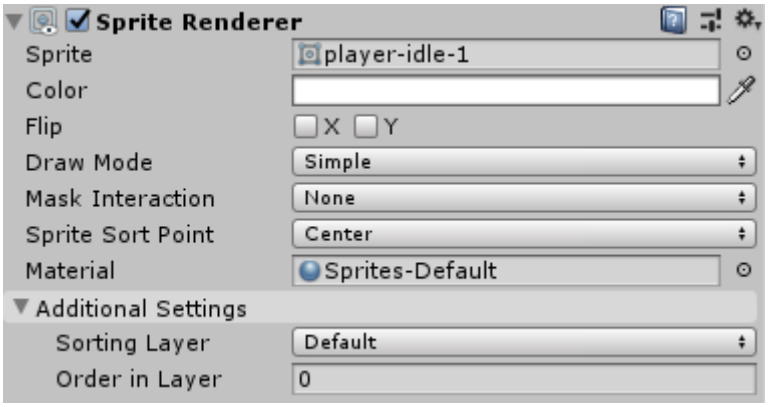

Рис.3. Вікно компоненти Sprite Renderer

Наступним етапом після додавання ігрового об’єкту є налаштування взаємодії між об’єктами. Цей процес виконується в два кроки:

1. Додавання фізики на об’єкт
2. Додавання способу зчитування взаємодій об’єктів при зіткненні.

В Unity за фізику відповідає компонента Rigidbody. Тут важливим моментом, є те що для 2D і 3D існують окремі компоненти які відповідають за фізику у відповідному просторі. Насправді єлиною різницею між Rigidbody 2D і 3D, є те фізика 2D об’єктів буде опрацьовуватись лише при русі вздовж осей X,Y а поворот відносно осі Z. Для створення 2D гри використовується компонента Rigidbody 2D.

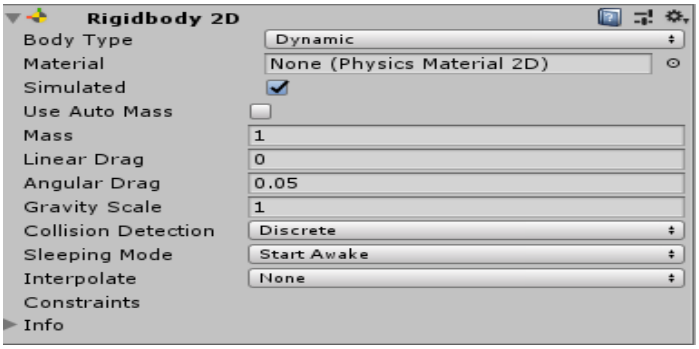

Рис.4. Вікно компоненти Rigidbody2D

Компонента Collider2D – відповаідає за зчитування взаємодій для ігрових об’єктів. При роботі з Rigidbody2D використовуються колайдери наступних типів:

- Circle Collider 2D
- Box Collider 2D
- Edge Collider 2D
- Polygon Collider 2D

Для роботи з ігровим персонажем в 2D, одною з найкращих практик є використання відразу 2х колайдерів(нижнє і верхнє тіло, як показано на Рис.5.

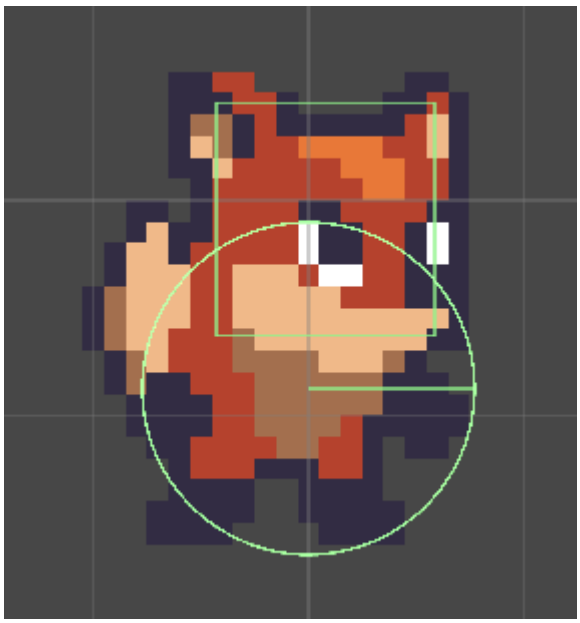

Рис.5. Конфігурація колайдерів на персонажі

2.6. Основи скриптування стану об’єктів.

Всі взаємодії в середині гри описуються за допомогою скриптів(коду написаного на мові C#). При створенні будь-якого скрипту в середині двигуна Unity ви отримаєте наступну структуру коду.

```
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        ...
    }
}
```

Як ми бачимо замість стандартних бібліотек мови C# ми маємо бібліотеку UnityEngine. Це фабрична бібліотека Unity з підтримкою основного робочого функціоналу. Також наші за замовчуванням будуть наслідуватись від класу MonoBehaviour, де описані основні методи для роботи з об'єктами на ігровій сцені.
Метод Start() є свого роду початком виконання програми. Те що розташоване в ньому відпрацює один раз після завершення загрузки сцени.
Метод Update() відпрацьовуватиме кожен кадр системи, що дозволяє зручно працювати з подіями які повинні відбуватись завжди(наприклад перевірка натиснення клавіши клавіатури).
В бібліотеці Unity реілзовано спеціальний тип даних для роботи з двовимірним простором Vector2 name – що аналогічно float[2]. При цьому прийнято що 1елемент такого вектору відповідає координаті X, другий Y.
Так само як працюючи з компонентами в ікні ієрархії ми можемо викликати їх зі скрипту, створивши екземпляр відповідного класу та давши йому силку на компоненту. Так як компонента Transform є обов'язковою компонентою будь-якого об'єкта на ігровій сцені, до неї реалізований скорочений механізм доступу. Так після підключення до об'єкту скрипта з наступним кодом:

```
Vector2 position;
void Start()
{
    transform.position = position;
}
```

Після запуску гри, об'єкт переміститься в позицію зі значеннями записаними в змінну position, яка в даному випадку дорівнює нулю.
Якщо перед Vector2 position додати модифікатор [SerializeField] ми отримаємо змогу керувати значеннями position з інспектора.
Так само через цей модифікатор ми можемо присвоювати силку на комоненту в інспекторі, наприклад при роботі з компонентою Rigidbody2D.
Тепер для реалізації руху ми можемо використати один з наступних методів:
transform.Translate(Vector2 translation) – приймає вектор2, по суті відстань і напрям зміни положення об'єкту.
Rigidbody2D.velocity = Vector2 velocity – встановлює швидкість об'єкту заданій в Vector2 velocity.
Rigidbody2D.AddForce(Vector2 force) – прикладає до об'єкту вибраний вектор сили.
Створвиши екземпляр обраної компоненти та кешувавши її ми можемо керувати її станом. Для кожної компоненти є свої властивості які ми можемо змінювати з скрипту. Наприклад положення X, Y, Z в компоненті трансформ чи Mass в компоненті Rigidbody2D. Також ми можемо використати розширюючий метод:

Component.enable = bool state – який виключить та включить обрану компоненту.

2.5. Створення відгуку при натисненні клавіш.

В Unity реалізована своя система Input, яка дозволяє отримувати відгук з різноманітних пристроїв керування таких як миш, клавіатура, джойстик та багато інших. В вікно Input можна потрапити через панель системних інструментів: Edit – Project Settings – Input(рис 6.).

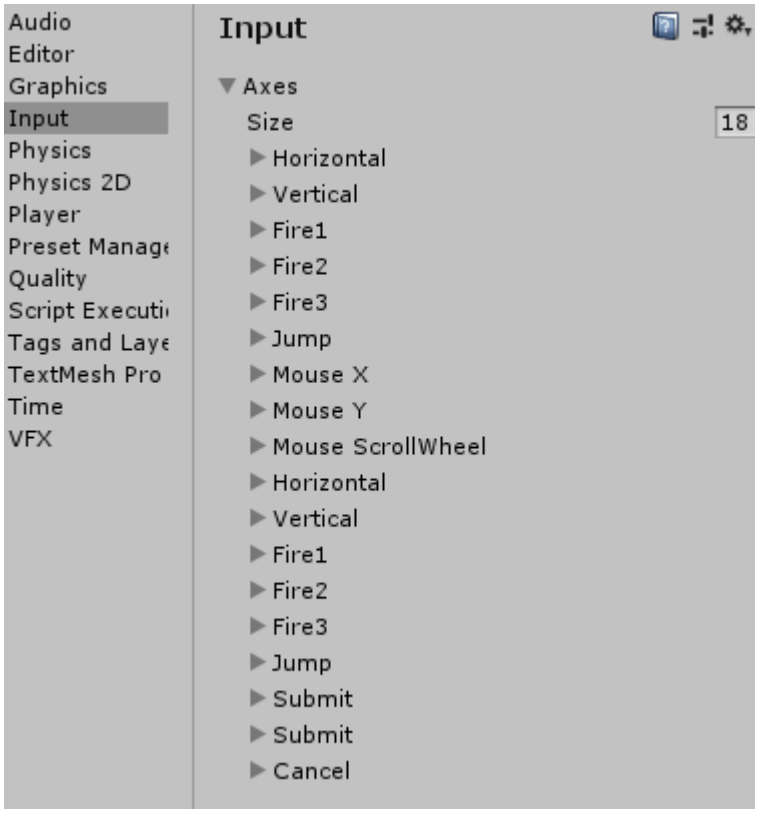

Рис.6 Панель Input в редакторі Unity

Тут знаходяться налаштовування фабричних назв клавіш для використання при скриптуванні.
При написанні скрипту, для зчитування вводу ми будемо використовувати методи з класу Input.
Input.GetAxis(string name); - приймає назву осі з вікна Input та повертає float в межах від -1 до 1. Наприклад вісь Horizontal зчитує натиснення клавіш (a,d, right, left) або горзонтальної осі джойтика і повертає -1 при лівих значеннях, 1 при правих. При чому це значення буде змінюватись плавно набуваючи десяткових та сотих значень в діапазоні (-1;1).
Input.GetAxisRaw(string name) – працює аналогічно до попереднього методу але повертає або -1 або 1.

Input.GetButton(string name) –приймає назву осі з вікна Input та повертає true при натисненні відповідної клавіши.
Input.GetKey(string name) – приймає системний клод клавіши і при її натисненні повертає true.

Список посилань:

1. Офіційна докумнетаціяя Unity. Режим доступу – [електронний ресурс] https://docs.unity3d.com
2. Мережа розробників Microsoft. Режим доступу – [електронний ресурс] https://msdn.microsoft.com/uk-ua/

Лабораторна робота №2
Робота з 2D анімацією

1. Постановка задачі

Налаштувати анімації бігу, спокою, стрибку та хотьби напресідки для персонажа з лабораторної роботи №1.

2. Мета роботи

1. Розуміння принципів анімації
2. Анімація об’єктів
3. Керування анімацією з скрипта.

3. Ключові положення

3.1 Принципи анімації.

Всього інсує 2грунтовних підходи, до виконання анімації в ігровому двигуні Unity. Перший це покадрова анімація. В основі цього принципу лежить створення зображень об’єкта на різних стадіях руху, та прокрутка цих зображень в заданому порядку. Приклад показано на рис.7.

Рис.7 Приклад покадрової анімації

Цей підхід є основним для створення більшості 2D анімації. Інший підхід в юніті базується на використанні додаткового програмного забезпечення Anima2D. В його основі є створення скелету для спрайта, та керування спрайтом через рух цього скелету.

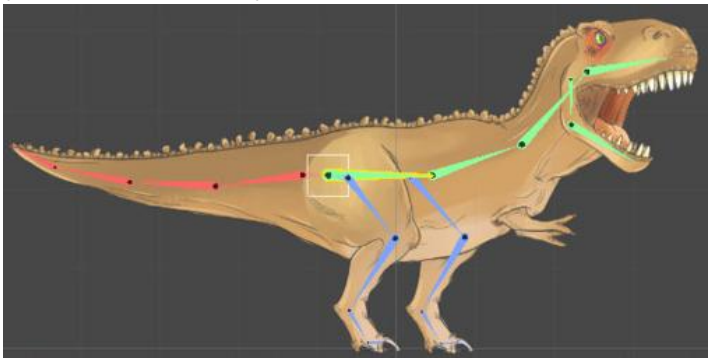

Рис.8. Приклад скелетної анімації в Unity

3.2 Анімація об’єктів

Анімація об’єктів в Unity відбувається в два кроки:

1. Створення анімаційних кліпів
2. Налаштування переходів між ними

Анімаційні кліпи створюються за допомогою вікна Animation(Window – Animation – Animation). Приклад вигляду такого вікна представлено на рис. 9.

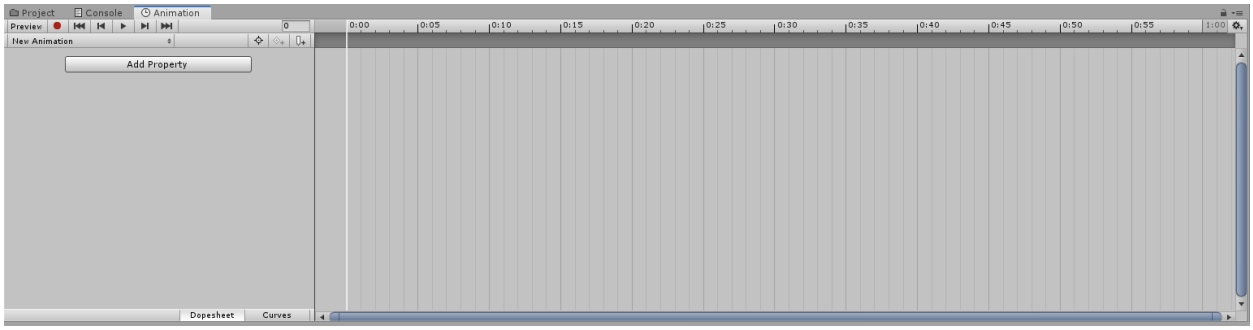

Рис.9 Вікно анімації

В цьому вікні ми можемо побачити часову шкалу, на якій за принципом Drug and Drop ми пириміщаємо спрайти для створення анімаційного кліпу. В разі використання скелетної анімації, ми можемо записувати зміну положення частин спрайту для отримання потрібного кліпу. Важливо, що кожен анімаційний кліп повинен відповідати за один стан об’єкту(наприклад біг, стан спокою, стрибок).

Після створення потрібної кількості анімаційних кліпів, ми повинні налаштувати переходи між ними. Для цього потрібно перейти в вікно Animator рисю 10(Window – Animation – Animator), який підключається до конкретного об’єкту як компонента.

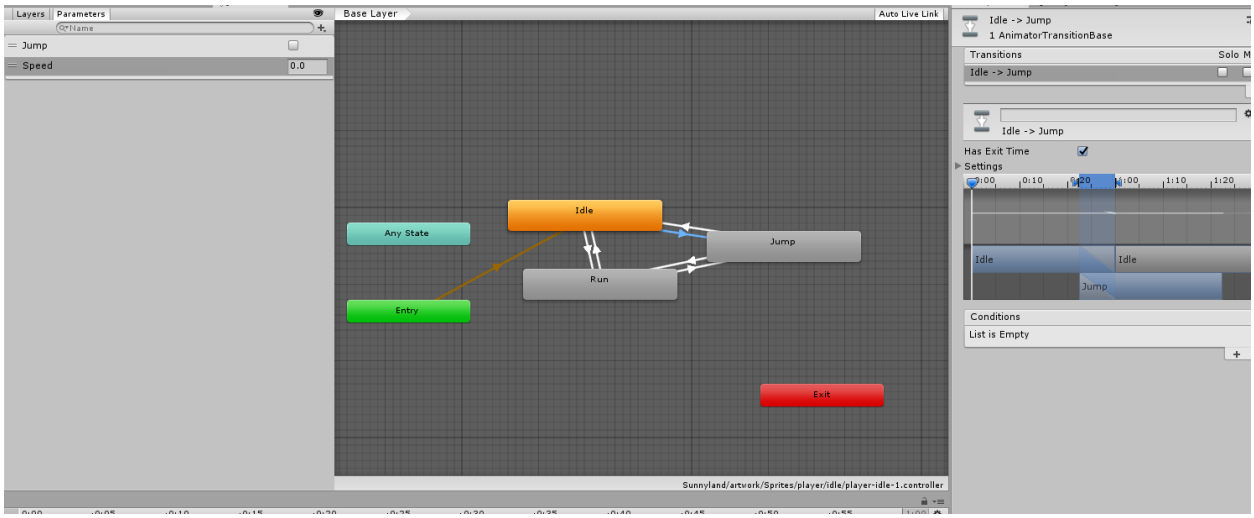

Рис.10 Вікно Animator

В даному вікні анімаційні кліпи представлені у вигляді діаграми, де помаранчевим позначено стан за заповчуванням. Переходи між станами виконуютья за допомогою прокладення зв’язків між елементами діаграми(білі стрілочки) та задання значень параметру, при зміні якого буде відбуватися конкретний перехід. В інспекторі кожного переходу ми задаємо умову зміни параметра при якому буде відбуватись перехід. Параметри можуть бути одного з чотирьох типів:

- Int
- Float
- Bool
- Trigger

3.3 Керування анімацією за допомогою скрипту

Для доступу до комоненти Animator нам потрібно створити екземпляр відповідного класу та кешувати його. Наприклад:
Animator animator;
Тепер в нас є доступ до ряду методів, які дозволяють нам змінювати значення параметрів в аніматорі:
animator.SetBools(string parametrName, bool value); - Метод приймає ім’я булівського параметру в Animatori та вситавляє відповідне значення.
animator.SetFloat(string parametrName, float value) - Метод приймає ім’я параметру типу float в Animatori та вситавляє відповідне значення.

Список посилань:

1. Офіційна докумнетаціяя Unity. Режим доступу – [електронний ресурс] https://docs.unity3d.com
2. Мережа розробників Microsoft. Режим доступу – [електронний ресурс] https://msdn.microsoft.com/uk-ua/

Лабораторна робота №3
Реалізація пострілів для 2D ігор

1. Постановка задачі

Реалізувати можливість стріляти для персонажа з лабораторної роботи №1.

2. Мета роботи

1. Розуміння принципів колізії.
2. Реалізація стрільби за допомогою префабів.
3. Реалізація стрільби за допомогою RayCast.

3. Ключові положення

3.1. Принципи колізії

Для реалізації колізії на всіх об’єктах учасниках колізії повинні буди присутні компоненти типу Collider. Після цього ми можемо контролювати наслідки колізії через скрипт. Для цього в Unity існує шість методів обробки колізії для 2D ігор:

private void OnCollisionEnter2D(Collision2D collision){} – спрацьовує один раз на початку зіткнення об’єкту з будь-яким іншим, дозволяючи отримати інформацію про нього.

private void OnCollisionExit2D(Collision2D collision){} - спрацьовує один раз в кінці зіткнення об’єкту з будь-яким іншим, дозволяючи отримати інформацію про нього.

private void OnCollisionStay2D(Collision2D collision){} - спрацьовує на початку зіткнення об’єкту з будь-яким іншим, дозволяючи отримати інформацію про нього.

private void OnTriggerEnter2D(Collider2D collision){} – спрацьовує один раз на початку зіткнення об’єкту з будь-яким іншим, дозволяючи отримати інформацію про нього, з тою різницею що об’єкт на якому висить скрипт перебуває в режимі isTrigger – зіткнення не відбуваються, але фіксуються.

private void OnTriggerExit2D(Collider2D collision){} - спрацьовує один раз в кінці зіткнення об’єкту з будь-яким іншим, дозволяючи отримати інформацію про нього, з тою різницею що об’єкт на якому висить скрипт перебуває в режимі isTrigger – зіткнення не відбуваються, але фіксуються.

private void OnTriggerStay2D(Collider2D collision){} - спрацьовує на початку зіткнення об’єкту з будь-яким іншим, дозволяючи отримати інформацію про нього, з тою різницею що об’єкт на якому висить скрипт перебуває в режимі isTrigger – зіткнення не відбуваються, але фіксуються.

3.2. Реалізація стрільби за допомогою префабів.

Префаб – це об’єкт з ігрової сцени збережений в вигляді файла, в директорії проекту. Він зберігає в собі всі налаштування, що ви зробили на ігровій сцені для об’єкта дозволяючи таким чином використовувати копії одного і того ж об’єкту. Це особливо зручно при побудові нових сцен та створенні об’єктів з коду.

Таким чином, один способів реалізації стрільби, є використання префабу кулі, якій в початковий момент часу надається швидкість в потрібному напрямку. Після цього всі подальші опрацювання зіткнення відбуваються в середині кулі.

Метод яким ми можемо створити копію об’єкту на ігровій сцені:
Instantiate(GameObject prefab, Transform position, Transform rotation) – створює копію обраного префабу у вказаній позиції та з вказаним кутом повороту.

3.3. Реалізація стрільби за допомогою RayCast.

Raycast - дозволяє нам створювати невидимі промені, які летять з умовоною швидкістю світла і дозволяють нам отримати інформацію про об’єкти з якими даний промінь стикається. Наприклад строчка коду
RaycastHit2D info = Physics2D.Raycast(Vetor2 point, Vector2 direction); - випускає промінь з точки point в напрямку direction. При цьому в змінну info, записується інформація про перший колайдер з яким зіткнеться наш промінь.

3.4. Реалізація системи нанесення шкоди

Тепер єдине що залишилось реалізувати систему нанесення шкоди. Для цього нам потрібно об’єднати групу об’єктів під потрібним тегом та прикріпити до них скрипти контролю. В скриптах нам буде достатньо одного публічного методу, який ми будемо викликати при зіткненні з променем/префабом.

Для цього ми можемо використати розширяючий метод:
GetComponent<Type>().Method(); - отримує з конкретного об’єкту(в нашому випадку колайдера з яким зіткнулась куля/префаб) компоненту заданого типу та виконує в ній обраний метод.

В методі ж нам буде достатньо написати рядок:
Destroy(gameObject); - метод знищує об’єкт на якому прикріплений даний скрипт.

Список посилань:

1. Офіційна докумнетацияя Unity. Режим доступу – [електронний ресурс] https://docs.unity3d.com
2. Мережа розробників Microsoft. Режим доступу – [електронний ресурс] https://msdn.microsoft.com/uk-ua/