



GAMELAB

GUIDEBOOK

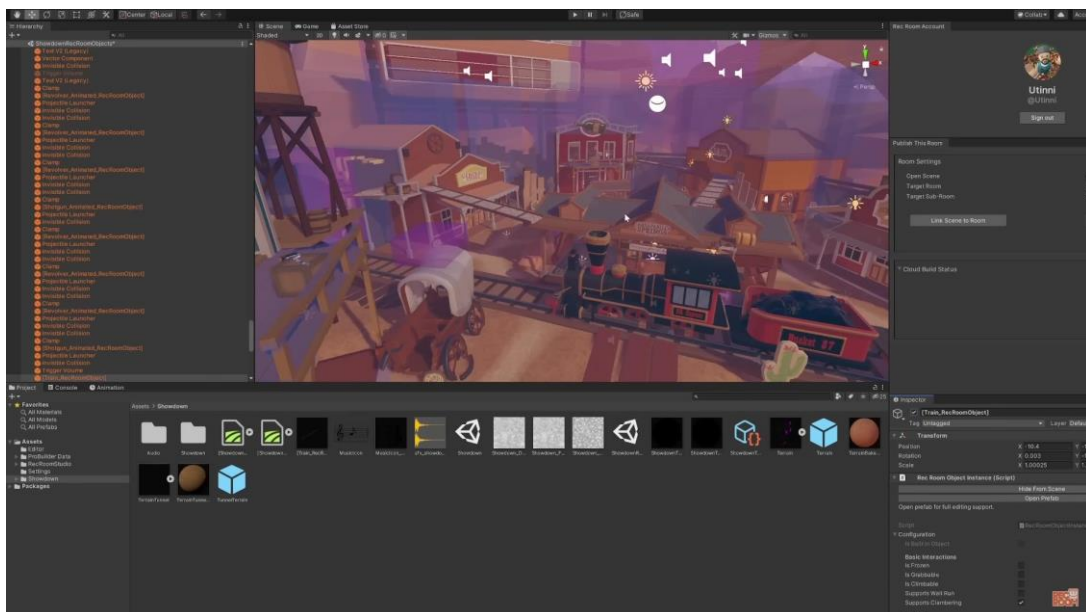
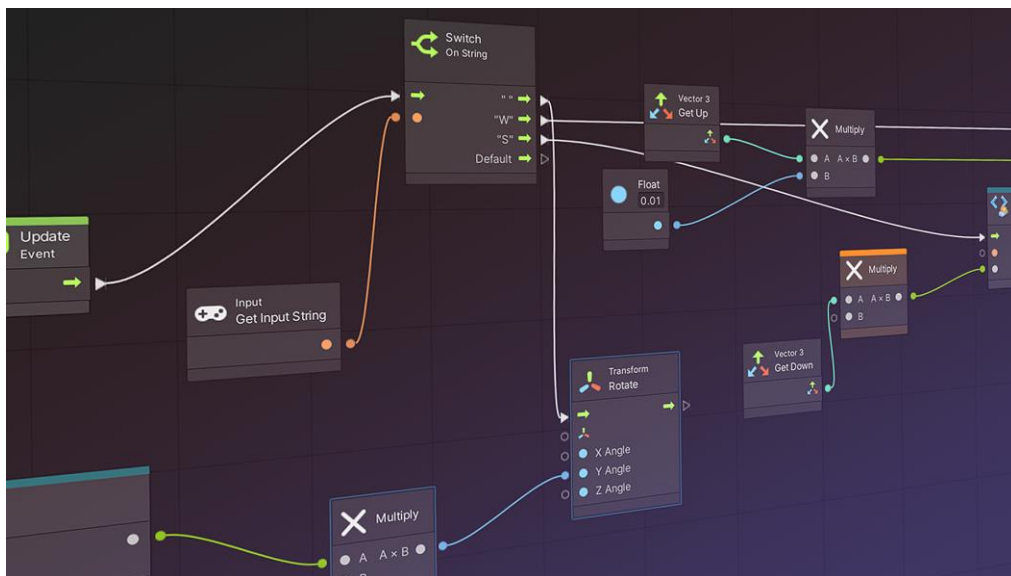
NAME

สารบัญ

รู้จัก Unity สักหน่อย	1
วิธีติดตั้ง Unity Hub	2
วิธีติดตั้ง Unity version 2021	3
Unity ของเรา	5
แล้วเราจะสร้าง Game Object ได้อย่างไร ?	7
Unity Asset	8
Import Unity Asset ยังไง?	9
Basic C# Programming	10
โครงสร้างภาษา C# ใน Unity	11
Variable	14
Public / Private	15
Math Operator	16
Boolean operator	18
If / Else operator	19
For Loop	20
Array	21
Unity Basic	22
Transform Parent and child	23
Sprite renderer	24
transform.Translate	25
Input.GetAxis	26
Vector3.Lerp	27
Random.Range	27
Mathf	28
Collider2D	29
Rigidbody2D	30
Collision ต่างกับ trigger ยังไง ?	32
ตัวอย่างคำสั่งในการใช้งาน	32
gameObject.tag	36
Destroy()	36
Prefab	37
Instantiate	37
การBuildเกมใน Unity	38

รู้จัก Unity สักหน่อย

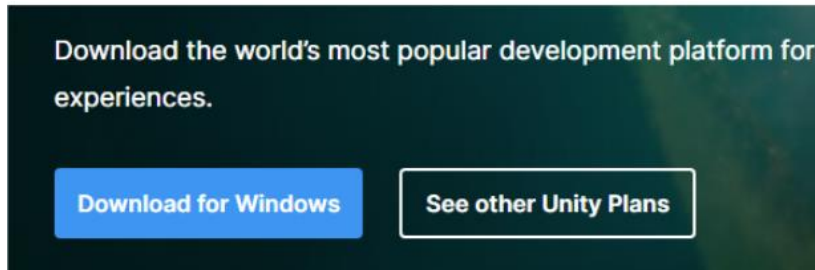
Unity เป็นเครื่องมือชนิดหนึ่งที่ใช้พัฒนาซอฟต์แวร์เกม (Game Engine) และงานด้านสื่อต่าง ๆ ตัว Unity เองได้มีการพัฒนา Unity Technology ให้สามารถนำไปใช้ในอุปกรณ์ได้หลากหลายโดยที่เห็นกันได้ทั่วไปคือ โทรศัพท์มือถือ(Android , IOS), Playstation, Xbox, เครื่องเล่น VR เป็นต้น



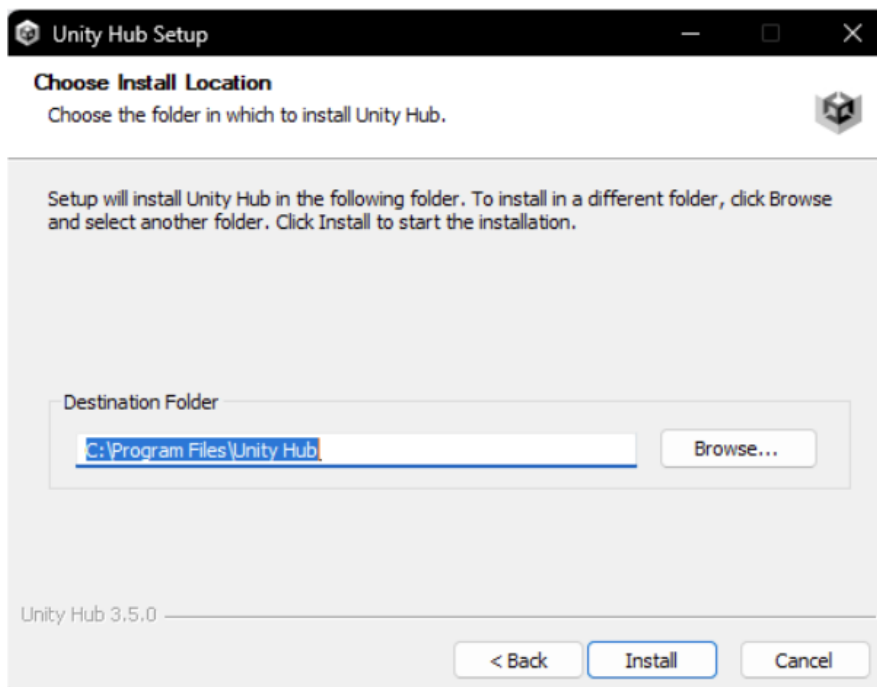
วิธีติดตั้ง Unity Hub

1.ดาวน์โหลด Unity hub จากลิงค์ต่อไปนี้ <https://unity.com/download>

2.คลิกที่ Download for Windows (ช่องสีฟ้า)

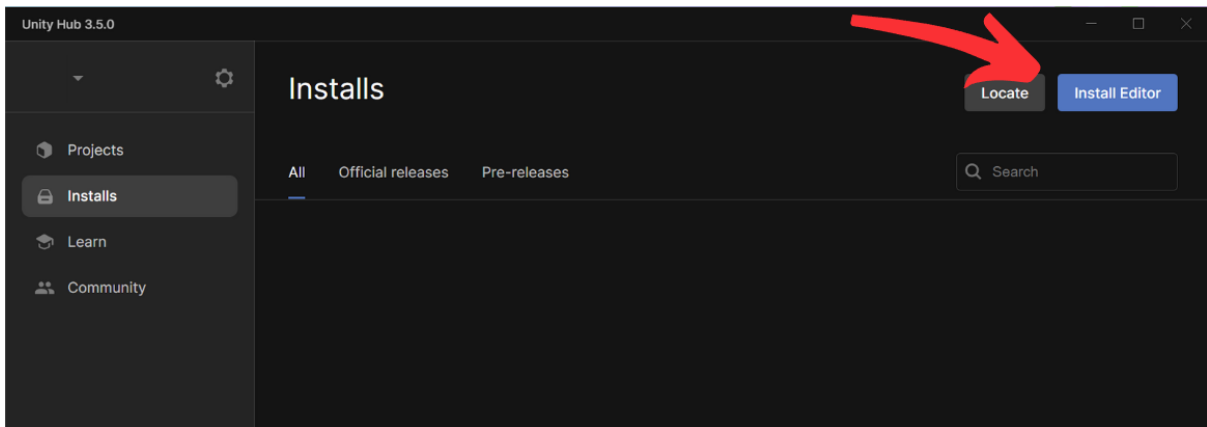
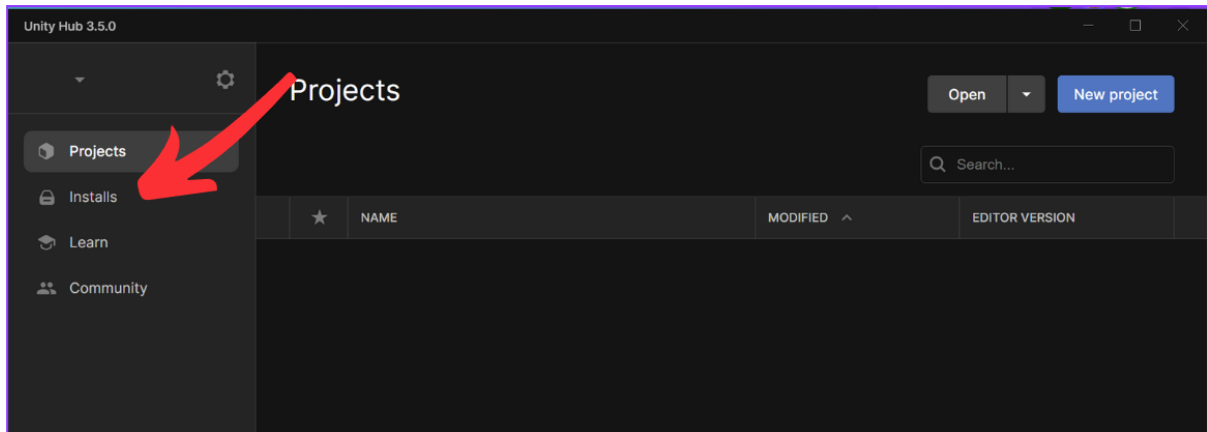


3.เปิดไฟล์ที่ดาวน์โหลดมาแล้วทำการติดตั้งตามขั้นตอนปกติ
หลังจากนั้นให้เลือกที่อยู่ของไฟล์ให้เรียบร้อยแล้วกด Install เป็นอันเสร็จสมบูรณ์

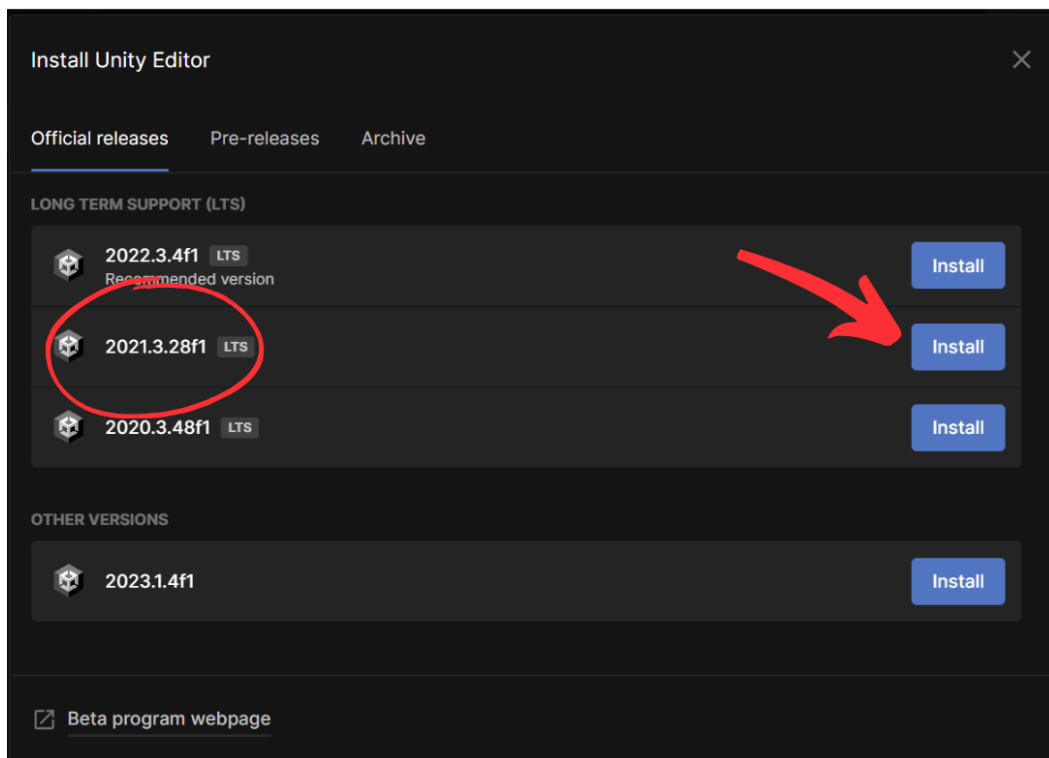


วิธีติดตั้ง Unity version 2021

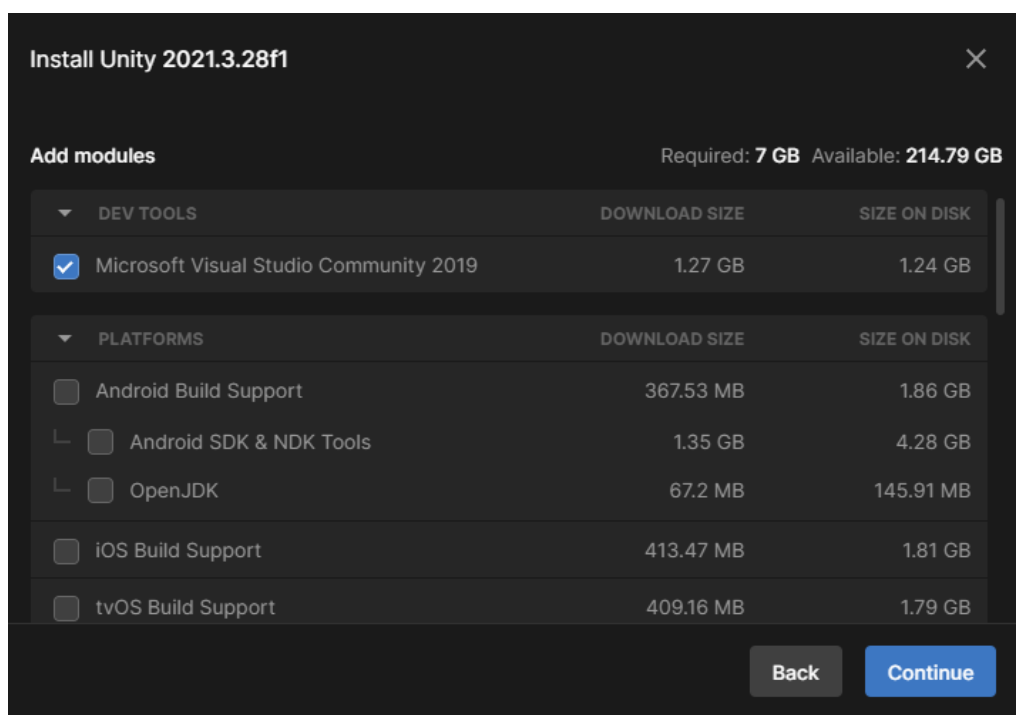
1.เมื่อเปิด **Unity Hub** ขึ้นมาให้คลิกที่ **Installs** จากเมนูด้านซ้าย จากนั้นเลือก **Install Editor**



2. มองหา **Version 2021.3.28f1** แล้วกด Install ทางขวา

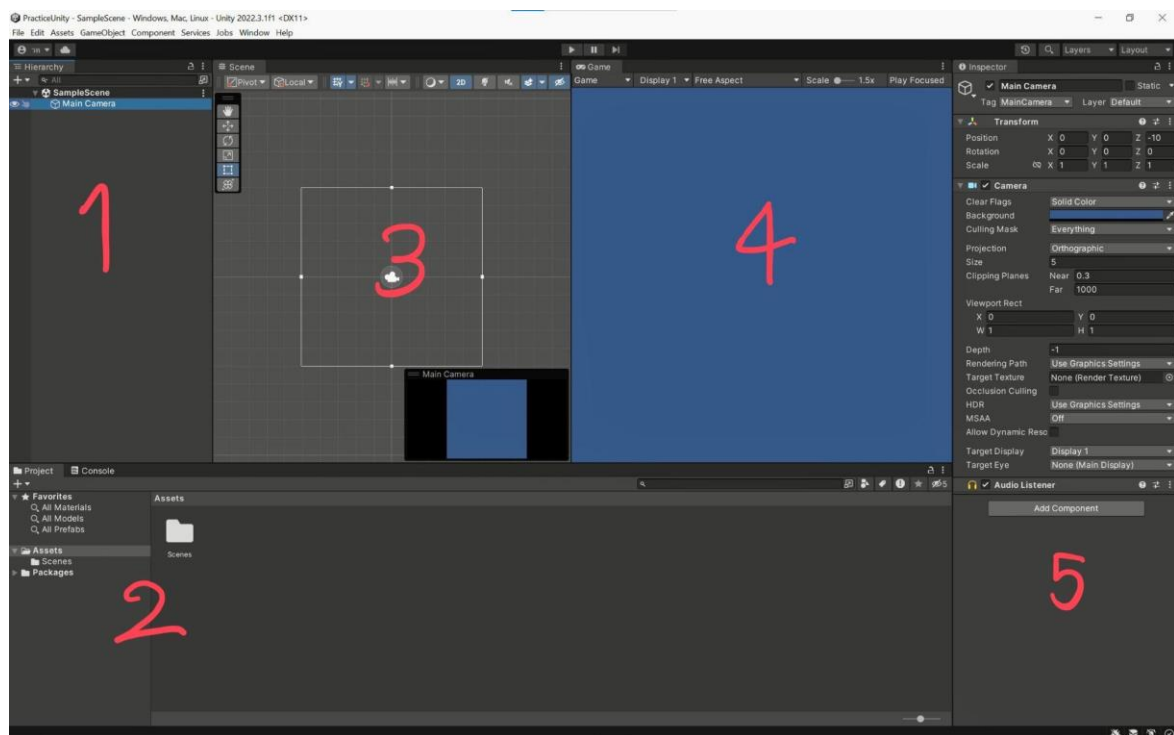



3. เมื่อนำหน้าต่าง Add Modules ขึ้นมา ให้คลิก **Continue**




Unity ของเรา

เรามาดู Layout(หน้าต่าง) ในโปรแกรมกันดีกว่า จะได้ไม่มึนหัว



(1) **Hierarchy(ไฮแรร์คี)**  **Hierarchy** : สำหรับสร้าง Game Object ต่างๆ ที่เราต้องการเช่น Camera , 2D/3D Object เป็นต้น และสามารถจัดการ Object ต่าง ๆ ในหน้า Scene




(2) **Project (โปรเจค)**  **Project** : ไว้ใช้เก็บและเรียกใช้ Asset(วัตถุดิบ)ต่างๆ ที่เราหามาได้จาก Unity Asset หรือที่เราทำขึ้นเอง

Console(คอนโซล)  **Console** เป็นเหมือนตัวแสดง Debug และ Error เมื่อเราทำบางอย่างผิดพลาดกับตัว Project ของเรา หรือแสดง Debug ที่เราตั้งไว้

(3) หน้าต่าง View Scene

:เป็นส่วนที่ใช้แสดงภาพองค์ประกอบต่าง ๆ ของ scene ใช้เลือกเคลื่อนย้ายแก้ไข Object ต่าง ๆ

(4) หน้าต่าง Game Scene

:เป็นหน้าจอที่ใช้สำหรับทดลองเล่นเกมที่เราสร้างขึ้นมา ใช้คู่กับ ปุ่ม Play 
/Pause  /Step  ใน Toolbar

(5) หน้าต่าง INSPECTOR

:เป็นหน้าต่างที่ใช้สำหรับแก้ไขรายละเอียดของ Object , Material , Texture สามารถสร้างและเพิ่ม Component ต่างๆ ให้ Object ใน Scene *Script ก็ถือว่าเป็นหนึ่งใน Component เช่นกัน*

Q = Pan หรือการขยับเพื่อดู Scene ในมุมมองที่เราต้องการ

W = Move ใช้ในการเคลื่อนย้าย Object ของ Scene

E = Rotate ใช้ในการหมุนองศาของ Object ได้

R = Scale ใช้ในการปรับขนาดของ Object

T = Rect Tool ใช้ในการปรับขนาด ขยับ หรือ หมุน UI Element ได้

F = Focus ใช้ในการหา Object ที่ต้องการ ใน Hierarchy ได้

บางครั้งเมื่อเราสร้างเกมไปสักพัก Object ในเกมของเราจะเยอะขึ้นเรื่อยๆ หรือบางทีอาจจะมีการซ้อนทับกันทำให้เราหาไม่เจอได้ใน View Scene

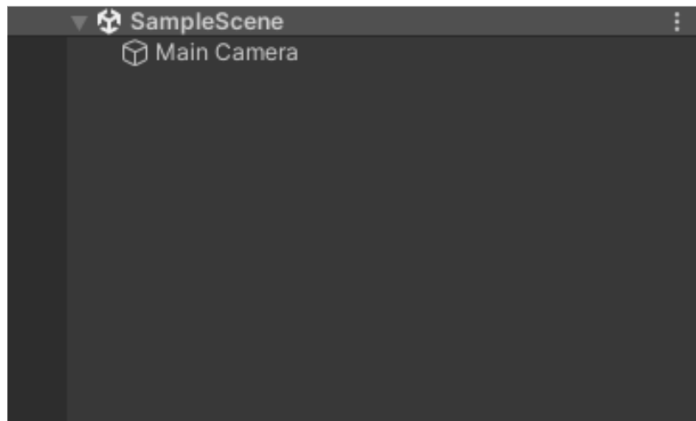
***UI คืออะไร - UI ย่อมาจาก User interfaces**

คือหน้าต่างข้อมูลที่ผู้เล่นเห็นเช่น จำนวนศัตรูบนหน้าจอ

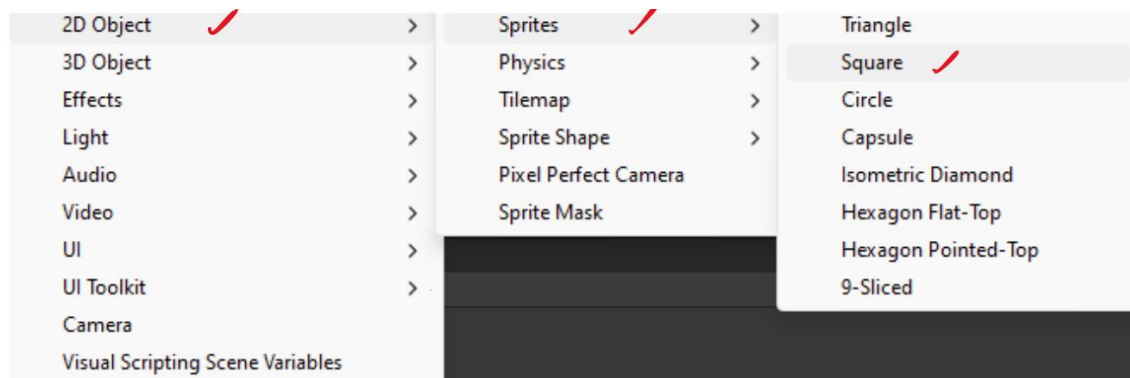
จำนวนเลือดของผู้เล่นหรือศัตรูเป็นสิ่งที่ใช้บอกที่กำลังเกิดอะไรขึ้นในเกมของเรา*

แล้วเราจะสร้าง Game Object ได้อย่างไร ?

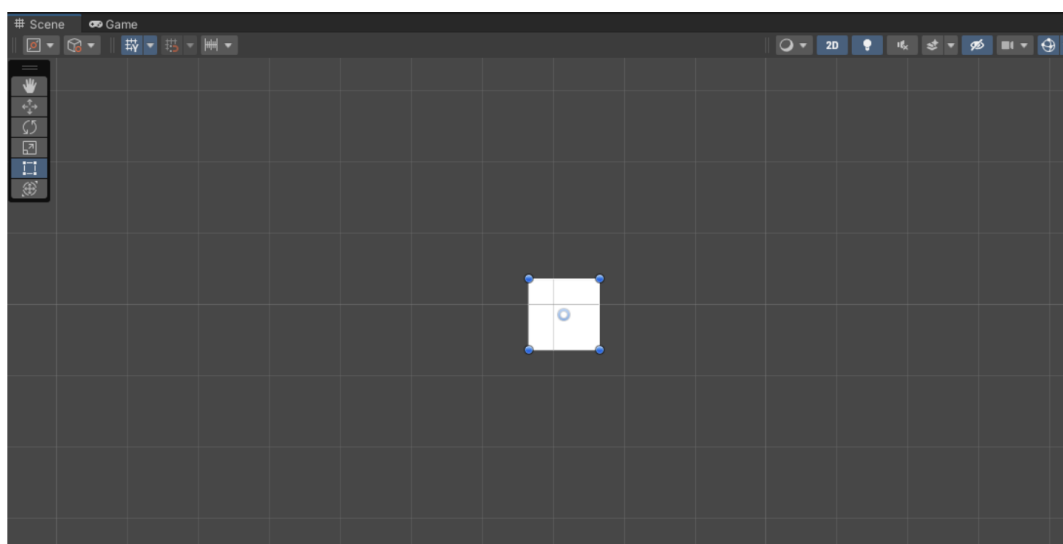
1. ในหน้า **Hierarchy** กดคลิกขวาที่ช่องว่าง



2. เลือกที่เราจะสร้าง Object อะไรก็ได้ ในที่นี้จะยกตัวอย่าง 2D object ใน Sprites เราสามารถสร้าง Shape ต่าง ๆ ออกมาได้

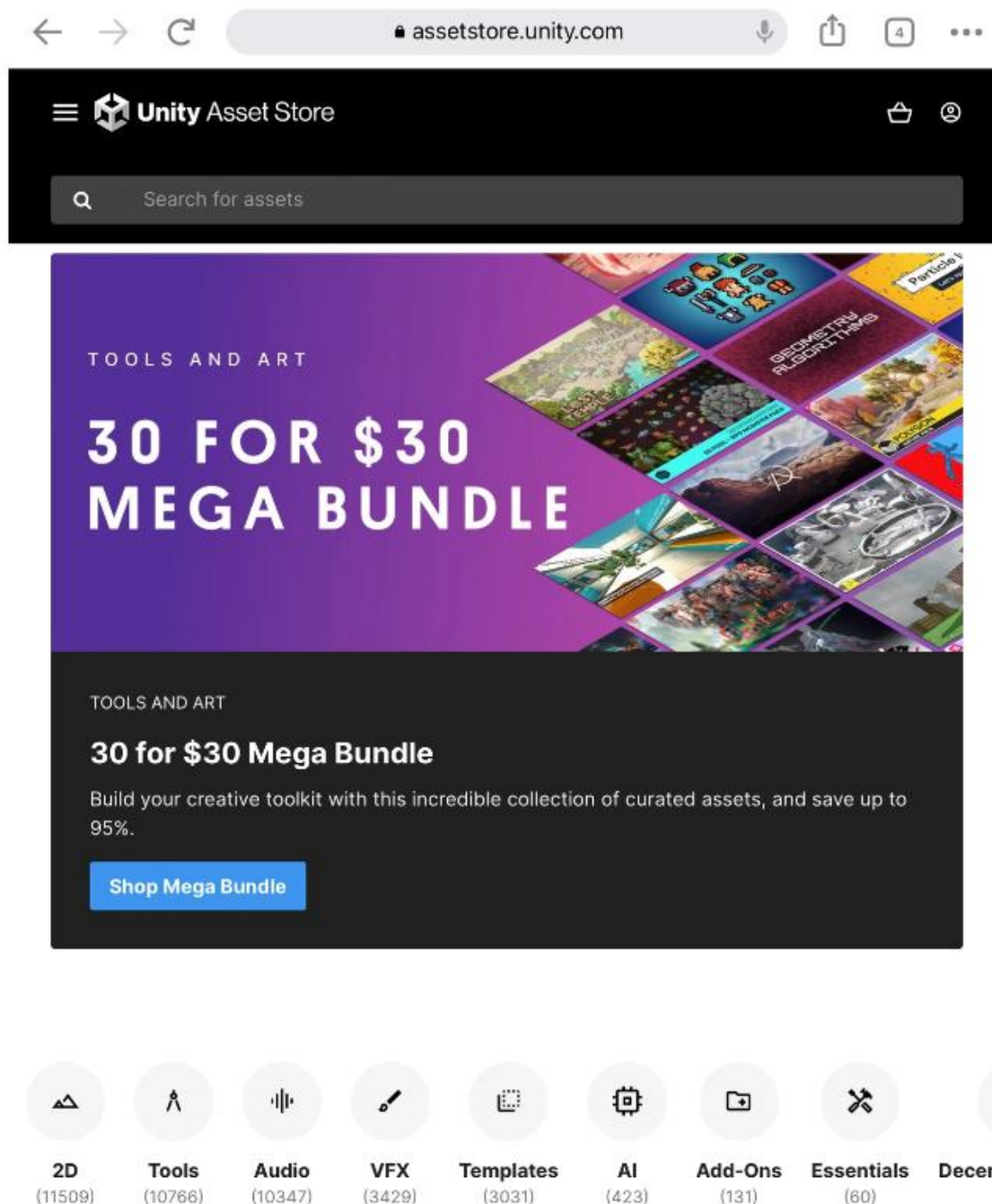


3. แล้วเราก็จะได้ Object Square ของเราแล้วในหน้า Scene



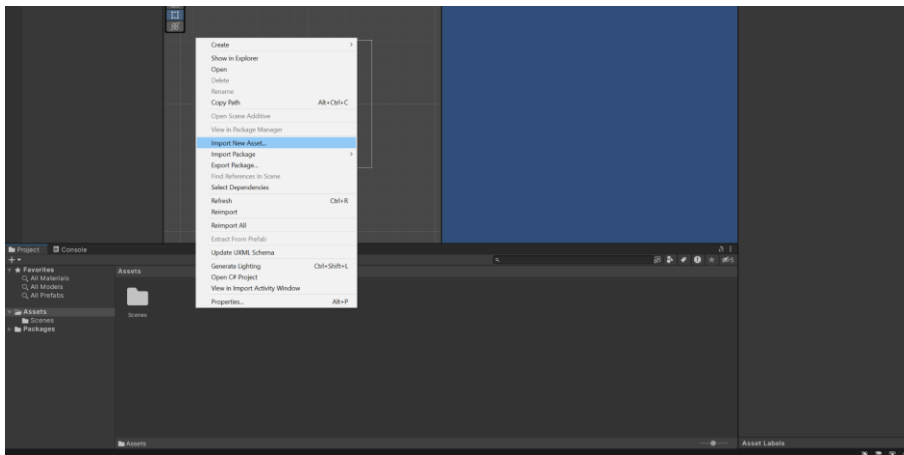
Unity Asset

Unity Asset เป็นที่รวมไฟล์เสริมเมื่อเราต้องการจะ Import สิ่งต่างๆ เช่น 2D/3D Graphic, 3D Model, VFX เป็นต้น เราสามารถเข้าไปที่เว็บของ **Unity Asset Store** เพื่อหาโหลดมาใส่ได้ในตัว Project ของเราได้อย่างอิสระ มีทั้งแบบฟรีและเสียเงิน

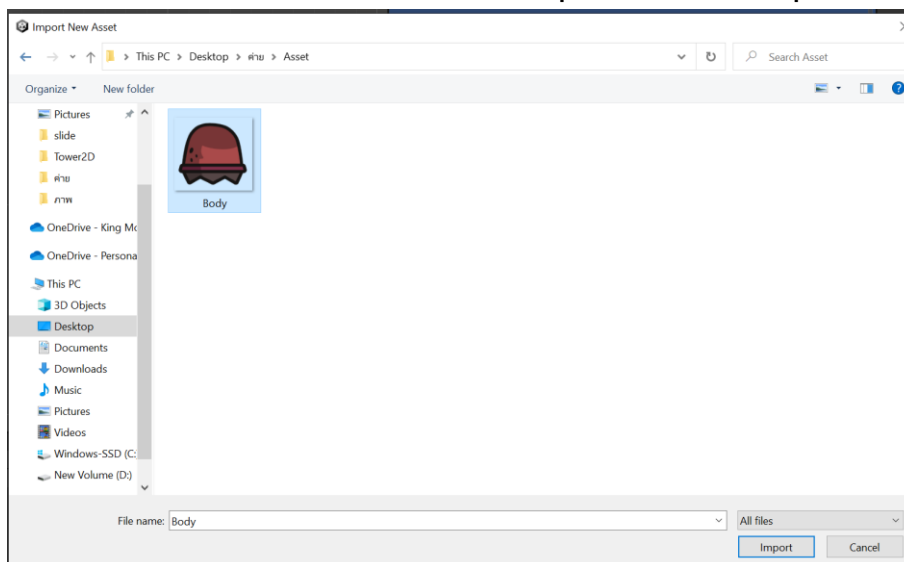


Import Unity Asset ยังไง?

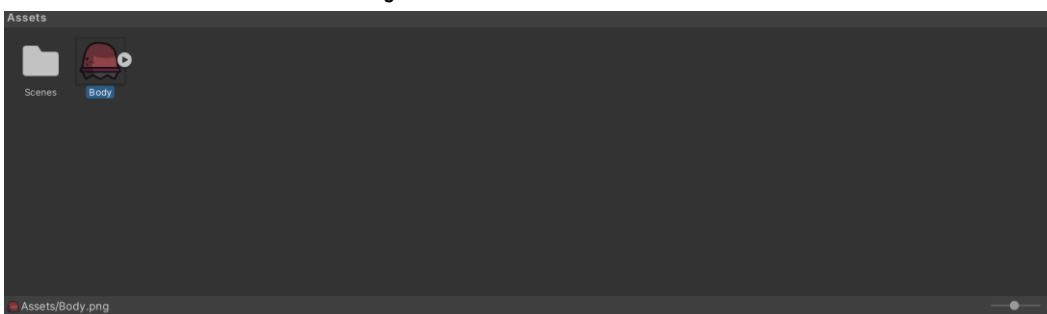
1.คลิกขวาตรงพื้นที่ว่าง Asset ใน Layout Project จากนั้นเลือกคำว่า Import New Asset



2.เลือก Asset ที่เราต้องการ Import แล้วกด Import ที่นี้ Asset ที่เรา



Import มากี่จะอยู่ในโฟลเดอร์ Asset ของเราแล้ว

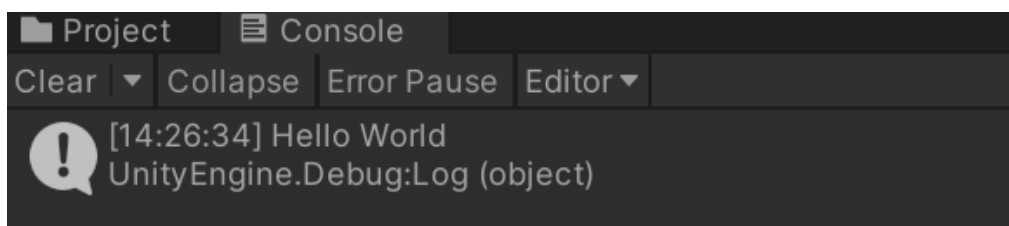


Basic C# Programming

C# คืออะไร C#(C Sharp) เป็นภาษาโปรแกรมที่พัฒนาขึ้นโดย Microsoft ใช้ในการพัฒนาแอปพลิเคชันต่าง ๆ บนแพลตฟอร์มต่าง ๆ และ .NET Framework เป็นระบบเฟรมเวิร์กที่ทำให้โปรแกรมเมอร์สามารถพัฒนาแอปพลิเคชันที่หลากหลายและทันสมัยได้อย่างสะดวกสบายมากขึ้น เราสามารถใช้ C# ในการกำหนดการทำงานของ Object ได้

#	Code
1	void Start()
2	{
3	Debug.Log("Hello World");
4	}

ตัวอย่างของ **Script Debug.Log**



เมื่อรัน (**Run**) แล้ว จะแสดงในหน้า **Console** แบบนี้

โครงสร้างภาษา C# ใน Unity

C# ใน Unity จะประกอบไปด้วยโครงสร้างส่วนต่าง ๆ ดังนี้

- Namespace
- Class
- Method

#	Code
1	<code>using System.Collections;</code>
2	<code>using System.Collections.Generic;</code>
3	<code>using UnityEngine;</code>
4	
5	<code>public class Program : MonoBehaviour</code>
6	<code>{</code>
7	<code> void start()</code>
8	<code> {</code>
9	<code> Debug.Log("Hello World");</code>
10	<code> }</code>
11	<code> void update()</code>
12	<code> {</code>
13	
14	<code> }</code>
15	<code>}</code>

Namespace คืออะไร - Namespace เปรียบเสมือนหนังสือหนึ่งเล่มซึ่งมีหลายๆ หน้าซึ่ง Class จะอยู่ภายใน Namespace เราใช้ using เป็นคำสั่งในการเรียกไลบรารี(Library) ที่มีความจำเป็นต่อการเขียนโปรแกรมภาษา C# หากไม่เรียกไลบรารีเข้ามาก็จะเริ่มต้นในการเขียนโปรแกรมไม่ได้

#	Code
1	using System.Collections;
2	using System.Collections.Generic;
3	using UnityEngine;
4	
5	public class Program : MonoBehaviour
6	{
7	void start()
8	{
9	Debug.Log("Hello World");
10	}
11	void update()
12	{
13	
14	}
15	}

Class คืออะไร - Class เปรียบเสมือนแบบแปลนในการสร้างวัตถุ เช่น Class ที่ชื่อว่า Car เราจะสามารถสร้างรถได้ ด้วยแบบแปลนนี้รถแต่ละคันอาจจะแตกต่างกันตรงคุณสมบัติต่าง ๆ ไม่ว่าจะเป็นสี, รูปร่าง, ยี่ห้อรถ เป็นต้น

การตั้งชื่อ Class ต้องขึ้นต้นด้วยตัวพิมพ์ใหญ่เสมอ

#	Code
1	using System.Collections;
2	using System.Collections.Generic;
3	using UnityEngine;
4	
5	public class Program : MonoBehaviour
6	{
7	void start()
8	{
9	Debug.Log("Hello World");
10	}
11	void update()
12	{
13	
14	}
15	}

ใน Script นี้จะแสดงข้อความ Debug.Log บน Console ว่า Hello World

Method คืออะไร Method ส่วนที่ใช้แสดงพฤติกรรมของ Object ที่ Script ได้ถูกใส่ไป Object นั้น ๆ เมื่อรันก็จะแสดงการทำงานเช่น เดิน, กิน, นอน เป็นต้น หรือตามที่เรต้องการเขียนไว้ว่าให้ Object นั้น ๆ ทำอะไร

Variable

ข้อมูล (Data) ในโปรแกรมมิ่งสามารถแบ่งประเภทได้หลายประเภท แต่ละประเภทมีลักษณะและการใช้งานที่แตกต่างกัน สำหรับภาษาโปรแกรม C# มีประเภทข้อมูลพื้นฐานที่สำคัญอยู่ 4 ประเภท ได้แก่ int, float, string และ bool หรือพูดง่ายๆ การกำหนดชนิดตัวแปรและค่าของตัวแปร

1. **int** หรือ **"integer"** : คือ ข้อมูลประเภทจำนวนเต็ม ทั้งเลขบวกและเลขลบ ไม่มีเลขทศนิยม เช่น -3, 0, 99

```
int myAge = 30;
```

2. **float** : คือ ข้อมูลประเภทจำนวนทศนิยม สามารถเก็บได้ทั้งเลขบวกและเลขลบ เช่น -3.14, 0.0, 99.99. ใน C# ต้องเติม **f** หรือ **F** ต่อท้ายค่าทศนิยมเช่น **-3.14f** , **99.99F** เป็นต้น

```
float myHeight = 1.8f;
```

3. **string** : คือ ข้อมูลประเภทข้อความ หรืออักขระต่อเรียงกัน ข้อมูลจะถูกเก็บไว้ในอัญประกาศ **"** เช่น "สวัสดีวันจันทร์" , "Xx_LnwZa007_xX"

```
string myName = Alice;
```

4. **bool** : หรือ **"boolean"** คือ ข้อมูลประเภทค่าความจริง ที่มีแค่สองค่าคือ **true** (จริง) หรือ **false** (เท็จ)

```
bool isOnline = true;
```

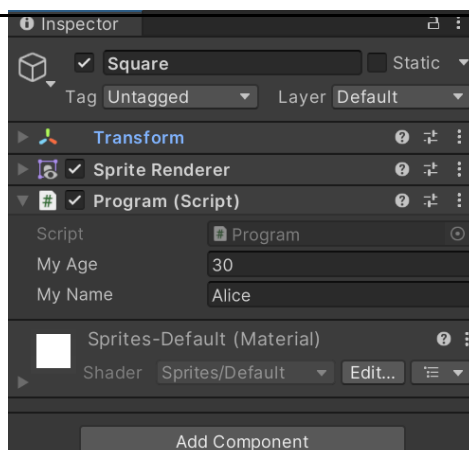
Public / Private

ใน Unity (และภาษา C# ทั่วไป) `public` และ `private` เป็นสิ่งที่เรียกว่า "access modifiers" หรือ ตัวกำหนดการเข้าถึง ซึ่งจะกำหนดว่าตัวแปรหรือฟังก์ชันสามารถเข้าถึงจากส่วนไหนของโค้ดได้บ้าง

Public : หากตัวแปรหรือฟังก์ชันถูกประกาศเป็น `public` มันจะสามารถเข้าถึงได้จากทุกส่วนในโปรแกรม ทั้งภายในคลาสเดียวกันและคลาสอื่น หรือจาก Unity Inspector โดยตรง นี่เป็นวิธีที่ดีในการตั้งค่าตัวแปรผ่าน Unity Editor

Private : แต่หากตัวแปรหรือฟังก์ชันถูกประกาศเป็น `private` มันจะสามารถเข้าถึงได้เฉพาะภายในคลาสที่ประกาศอยู่ ตัวแปรหรือฟังก์ชันส่วนตัวจะไม่สามารถเข้าถึงได้จากคลาสอื่นหรือ **Unity Inspector**

#	Code
1	public class Program : MonoBehaviour
2	{
3	public int myAge = 30;
4	private float myHeight = 1.8f;
5	public string myName = "Alice";
6	bool isOnline = true;
7	void start()
8	{
9	Debug.Log("Hello World");
10	}
11	}



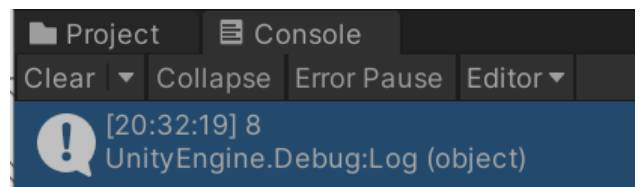
Math Operator

ในภาษา C# (และใน Unity) มี operator

พื้นฐานสำหรับการคำนวณทางคณิตศาสตร์ (Math operators) ได้แก่ +, -, *, /, %

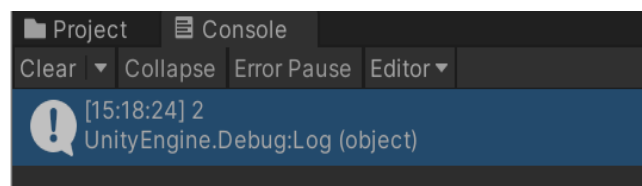
1. '+' : ใช้สำหรับการบวกค่าของตัวแปรหรือค่าคงที่

#	Code
1	void Start()
2	{
3	int a = 5;
4	int b = 3;
5	Debug.Log(a + b);
6	}



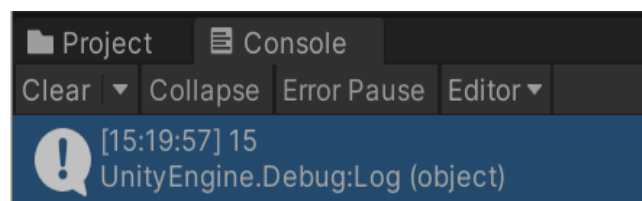
2. '-' : ใช้สำหรับการลบค่าของตัวแปรหรือค่าคงที่

#	Code
1	void Start()
2	{
3	int a = 5;
4	int b = 3;
5	Debug.Log(a - b);
6	}



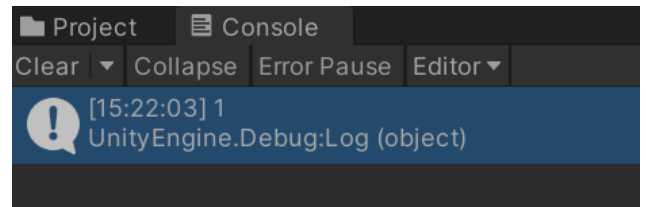
3. '*' : ใช้สำหรับการคูณค่าของตัวแปรหรือค่าคงที่

#	Code
1	void Start()
2	{
3	int a = 5;
4	int b = 3;
5	Debug.Log(a * b);
6	}



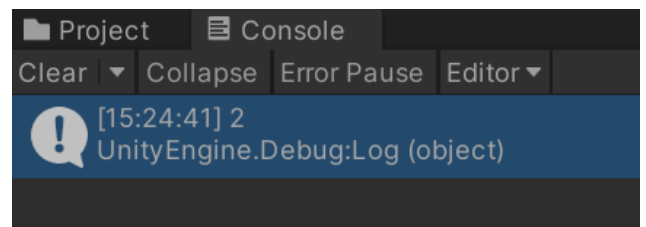
4. ``/`` : ใช้สำหรับการหารค่าของตัวแปรหรือค่าคงที่ ตัวอย่าง $5/3 = 1$ เศษ 2
เอาแค่ 1 มาคำนวณ

#	Code
1	<code>void Start()</code>
2	<code>{</code>
3	<code>int a = 5;</code>
4	<code>int b = 3;</code>
5	<code>Debug.Log(a / b);</code>
6	<code>}</code>



5. ``%`` : ใช้สำหรับการหาเศษของการหารค่าของตัวแปรหรือค่าคงที่ $5/3 = 1$ เศษ 2
เอาแค่ 2 มาคำนวณ

#	Code
1	<code>void Start()</code>
2	<code>{</code>
3	<code>int a = 5;</code>
4	<code>int b = 3;</code>
5	<code>Debug.Log(a % b);</code>
6	<code>}</code>



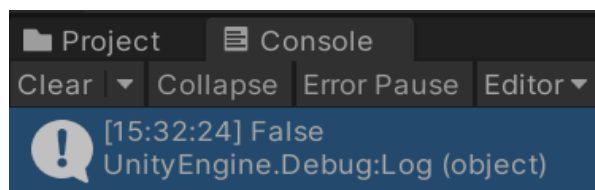
ให้จำไว้ว่าการคำนวณทางคณิตศาสตร์จะมีการใช้ "Operator Precedence"
ที่คล้ายกับการคำนวณคณิตศาสตร์ทั่วไป ดังนั้น `*`, `/`, `%` จะได้รับการดำเนินการก่อน `+`, `-`
ซึ่งคุณสามารถใช้วงเล็บ `()` เพื่อกำหนดลำดับการดำเนินการ

Boolean operator

Operator ทางตรรกะหรือ Boolean Operators ในภาษา C# (และใน Unity) ทำให้เราสามารถเปรียบเทียบค่าของตัวแปรหรือค่าคงที่และดำเนินการตามเงื่อนไขได้ มีประเภทหลัก ๆ คือ && (AND), || (OR), และ ! (NOT)

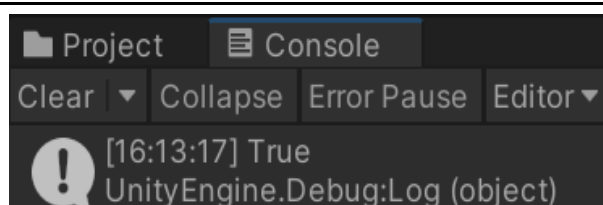
1. **&& (AND, และ):** จะให้ค่า true ถ้าทั้งสองฝั่งของ operator มีค่าเป็น true

#	Code
1	void Start()
2	{
3	bool a = true;
4	bool b = false;
5	bool result = a && b ;
6	Debug.Log(result);
7	}



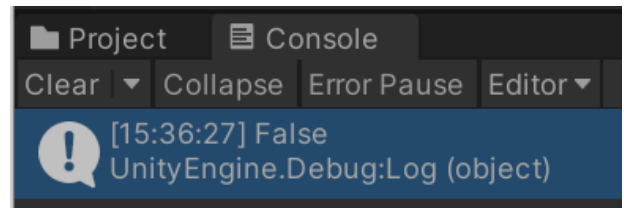
2. **|| (OR, หรือ) :** จะให้ค่า true ถ้าเข้าเงื่อนไขฝั่งใดฝั่งหนึ่งของ operator มีค่าเป็น true

#	Code
1	void Start()
2	{
3	bool a = true;
4	bool b = false;
5	bool result = a b ;
6	Debug.Log(result);
7	}



3. **!(NOT, นิเสธ)** : จะกลับค่าของ boolean นั้น ๆ ถ้าเป็น true จะกลับเป็น false และถ้าเป็น false จะกลับเป็น true

#	Code
1	void Start()
2	{
3	bool a = true;
4	bool result = !a ;
5	Debug.Log(result);
6	}



ยังมี Operator ที่ใช้ในการเปรียบเทียบด้วย ได้แก่ == (เท่ากับ), != (ไม่เท่ากับ), < (น้อยกว่า), > (มากกว่า), <= (น้อยกว่าหรือเท่ากับ), >= (มากกว่าหรือเท่ากับ) ซึ่ง operator เหล่านี้จะให้ค่า boolean (true หรือ false) ในการเปรียบเทียบค่าของตัวแปร ตัวอย่างเช่น 5 > 3 จะให้ค่าเป็น true

If / Else operator

If , Else คือคำสั่งสำหรับการสร้างเงื่อนไขแบบทางเลือก

If : If เป็นการสร้างเงื่อนไขแบบทางเลือกเดียว

else : else ใช้เขียนต่อจาก if สำหรับการทำงานคำสั่งใดๆ เมื่อเงื่อนไขเป็นเท็จ

#	Code
1	void start()
2	{
3	int age = 24;
4	if (age >= 18)
5	{
6	Debug.Log("You are an adult");
7	}
8	else
9	{
10	Debug.Log("You are an minor");
11	}
12	}

For Loop

การวนซ้ำ (Loop)

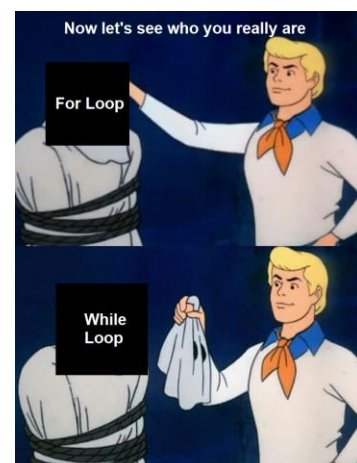
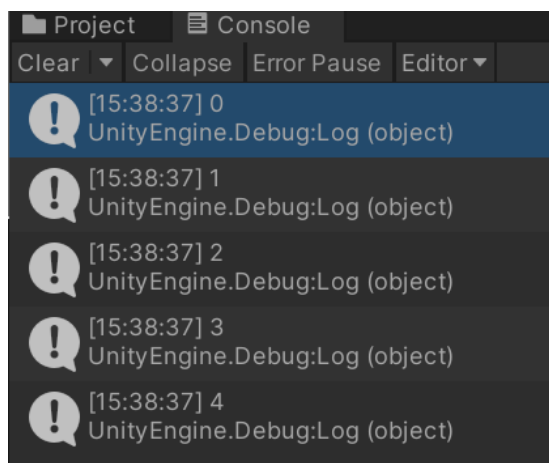
เป็นส่วนสำคัญของการเขียนโปรแกรมซึ่งช่วยให้เราสามารถรันคำสั่งหนึ่งหรือหลายคำสั่งซ้ำ ๆ จำนวนกี่ครั้งก็ได้ตามเงื่อนไขที่ใส่ไป

#	Code
1	void Start()
2	{
3	for(int i = 0; i < 5; i++)
4	{
5	Debug.Log("Hello World");
6	}
7	}

ในตัวอย่างนี้:

- $i = 0$ คือการกำหนดค่าเริ่มต้น
- $i < 5$ คือเงื่อนไขที่จะให้ loop ดำเนินการต่อ (ซึ่งจะวนซ้ำตราบใดที่ i ยังมีค่าน้อยกว่า 5)
- $i++$ คือการเพิ่มค่า i ขึ้น 1 หลังจากทุกครั้งที่วนซ้ำ
- `Debug.Log(i);` คือคำสั่งที่จะถูกดำเนินการในแต่ละการวนซ้ำ ซึ่งในกรณีนี้จะแสดงค่าของ i บน console ของ Unity

ผลลัพธ์ที่จะได้จาก for loop นี้คือจะแสดงเลข 0 ถึง 4 บน console ของ Unity



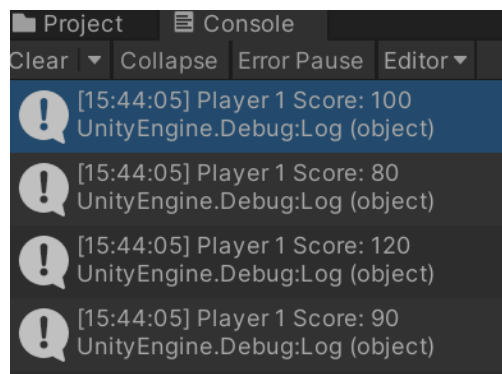
Array

อาร์เรย์เป็นโครงสร้างข้อมูลที่ใช้เก็บค่าประเภทเดียวกันได้หลายค่าในตัวแปรเดียว อาร์เรย์มีประโยชน์เมื่อเราต้องการเก็บชุดข้อมูลเดียว เช่น คะแนนของผู้เล่น ตำแหน่งของวัตถุในเกม หรือ ช่องเก็บของในเกม

#	Code
1	void Start()
2	{
3	int[] playerScores = new int[4];
4	playerScores[0] = 100;
5	playerScores[1] = 80;
6	playerScores[2] = 120;
7	playerScores[3] = 90;
8	Debug.Log("Player 1 Score: " + playerScores[0]);
9	Debug.Log("Player 1 Score: " + playerScores[1]);
10	Debug.Log("Player 1 Score: " + playerScores[2]);
11	Debug.Log("Player 1 Score: " + playerScores[3]);
12	}

ในตัวอย่างนี้:

- `int[] playerScores = new int[4];` คือการกำหนดขนาดของตัวแปร `playerScores` ซึ่งมีค่าเท่ากับ 4
- `playerScores[0, 1, 2, 3]` คือการกำหนดว่าในแต่ละช่องของ Array นั้นมีค่าเท่ากับเท่าไร



การนับของ Array นั้นต้องนับจาก 0 เสมอเพราะ 0 คือตำแหน่งที่ 1

Unity Basic

Get Component คือการดึงข้อมูลจาก Component อื่น ๆ ที่อยู่ใน Object ที่เราใส่ Script ไว้ทำให้เราสามารถเข้าถึงข้อมูลต่าง ๆ ใน Object นั้น ๆ ได้เช่น Transform, Rigidbody, Collider เราสามารถนำข้อมูลเหล่านั้นมากำหนดการทำงานของ Object นั้น ๆ ได้

#	Code
1	void Start()
2	{
3	HingeJoint hinge = GetComponent<HingeJoint>();
4	hinge.useSpring = false;
5	}

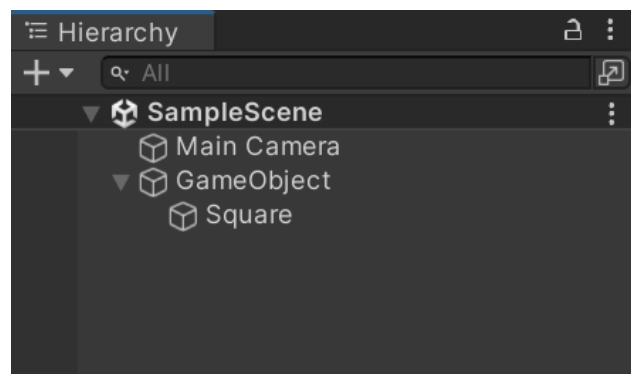
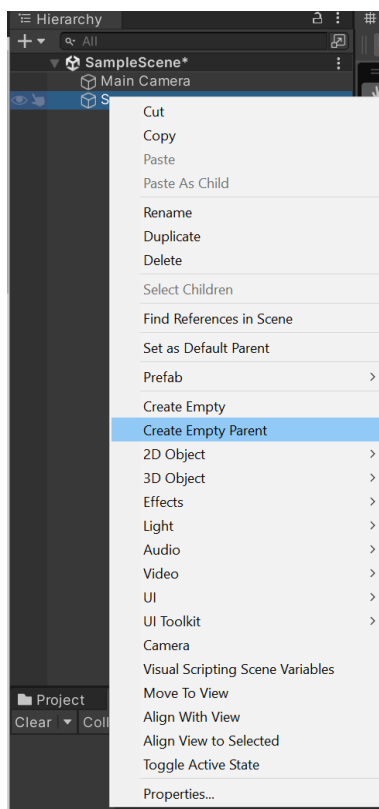
- โดยตัวอย่างข้างต้น มีการเรียก Component ที่มีชื่อว่า HingeJoint ใน Object ที่แปะ ScriptExample เอาไว้ และมีการตั้งชื่อตัวแปรมาเก็บค่าชื่อ hinge
- เครื่องหมายเท่ากับ คือ การกำหนดให้เก็บค่าตัวแปร ใน hinge
- คำสั่ง GetComponent<ใส่ชื่อไฟล์ที่ต้องการรับค่า>(); ใช้ในการรับค่าข้ามไฟล์ในที่นี้คือ HingeJoint

Transform Parent and child

ใน Unity นั้น Transform คือคลาสที่เก็บข้อมูลเกี่ยวกับตำแหน่ง(position), การหมุน (rotation) และขนาด (scale) ของ GameObject. นอกจากนี้ Transform ยังมีคุณสมบัติสำคัญที่ชื่อว่า Parent-Child Relationships หรือความสัมพันธ์ระหว่าง Parent (ผู้ปกครอง) และ Child (ลูก)

- **Parent** : คือ GameObject ที่อยู่ระดับบน หรือ GameObject ที่เป็น "ผู้ปกครอง" ของ GameObject อื่น ๆ
- **Child** : คือ GameObject ที่อยู่ภายใต้ Parent หรือ GameObject ที่เป็น "ลูก" ของ Parent GameObject

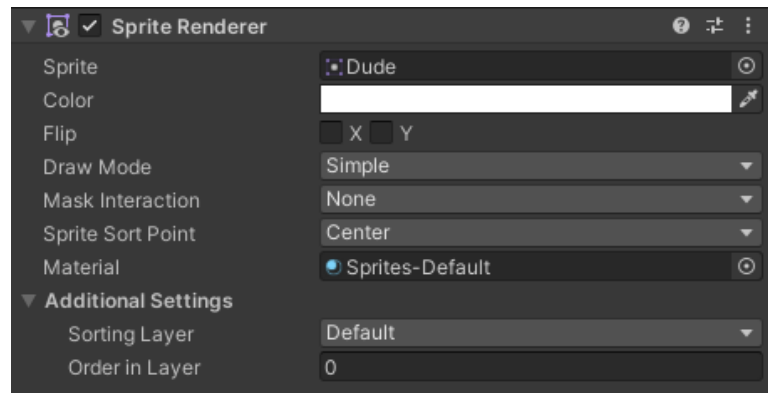
วิธีการสร้าง Parent คือ คลิกขวา Object ที่ต้องการให้เป็น Child + Create Empty Parent



ข้อสำคัญในการใช้ Parent-Child Relationships คือ การใช้ให้ถูกต้องสามารถทำให้การจัดการกับการเคลื่อนไหว และการจัดระเบียบ GameObject ใน Scene ของคุณง่ายขึ้น. ตัวอย่างเช่น คุณสามารถสร้าง GameObject ที่เป็น "Car" และมี GameObject อื่นๆ ที่เป็น "Wheel" เป็นลูกของ "Car" เมื่อคุณขยับ "Car" ไปที่ใดที่หนึ่ง "Wheel" จะตามมาด้วย.

Sprite renderer

เป็น Component ที่เราใช้แสดงภาพ 2 มิติได้

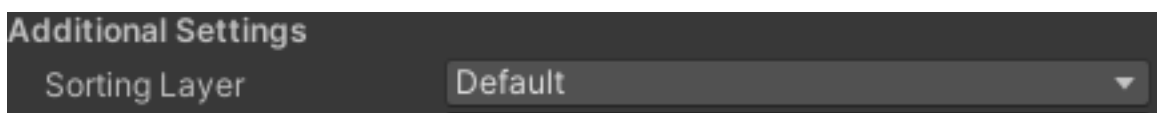


Sprite: ใช้สำหรับแสดงรูปภาพ 2 มิติที่อยู่ใน project ของเรา

Color: เราสามารถเปลี่ยนสีของ Sprite ได้
โดยการเปลี่ยนสีที่นี้จะเปลี่ยนสีโดยรวมของตัว Sprite



Flip: ใช้สำหรับกลับด้านของรูปภาพได้



Sorting layer: ใช้ในการเรียงลำดับรูปภาพได้เราสามารถตั้งชื่อให้กับ Layer เหล่านั้นถ้ามีเลขมากก็จะอยู่หน้าเลขที่น้อยกว่าและหาวัตถุเหล่านั้นอยู่ใน Sorting layer เดียวกัน เรายังสามารถกำหนดใน **Order in layer** (ลำดับในเลเยอร์) ได้อีกด้วย



tranform.Translate

ใน unity การที่เราจะเคลื่อน object ของเราด้วยคำสั่งนี้ **transform.Translate()** โดยเราสามารถกำหนดทิศทางต่าง ๆ ได้ด้วย

#	Code
1	<code>void Start()</code>
2	<code>{</code>
3	<code> transform.Translate(Vector3.up * Time.deltaTime);</code>
4	<code>}</code>

- **Vector3.up = Vector(0, 1, 0)**
- **Vector3.up** คือการให้วัตถุเคลื่อนไป 1 หน่วยในแนวแกน Y (ขึ้นด้านบน)
- **Time.deltaTime** ใช้ในการคำนวณว่า 1 เฟรมนั้น Object จะใช้เคลื่อนที่กี่ Unit ในหน่วยวินาที
- ใช้เพื่อให้ใน 1 วินาทีมีการอัปเดตอย่างมั่นคง



Input.GetAxis

ใน unity

เมื่อเราต้องการที่จะควบคุมตัวละครในเกมเราก็ต้องมีคำสั่งเช่นกันโดยที่นี้เราจะใช้

Input.GetAxis()

โดยเราจากอิงจากทิศทางการกดของผู้เล่นว่าเป็นแนวตั้ง("Vertical")

หรือแนวนอน("Horizontal")

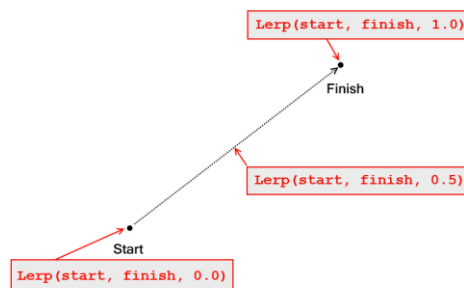
#	Code
1	public class PlayerController : MonoBehaviour
2	{
3	float moveSpeed = 5f;
4	void update()
5	{
6	float horizontalInput = Input.GetAxis("Horizontal");
7	float verticalInput = Input.GetAxis("Vertical");
8	Vector2 movement = new Vector2(horizontalInput, verticalInput);
9	movement.Normalize();
10	transform.Translate(movement * moveSpeed * Time.deltaTime);
11	}
12	}

- ทำการกำหนดความเร็วผ่าน float moveSpeed = 5f;
- ประกาศ 2 ตัวแปร horizontalInput และ verticalInput เพื่อเก็บค่าทิศทางการกดปุ่มผู้เล่นผ่านคำสั่ง Input.GetAxis("Horizontal")และ Input.GetAxis("Vertical")
- เมื่อผู้เล่นกดปุ่ม WASD หรือ Arrow key ไปในทิศทางใด object ก็จะเคลื่อนไปทางนั้น
- Vector2 คือคำสั่งที่ใช้ในการแก้ไขแกน X และ Y
- เราเรียกใช้คำสั่ง movement.Normalize() เพื่อให้ความเร็วนั้นสม่ำเสมอได้ และใช้ method Normalize() เพื่อให้ความเร็วในแนวทแยงไม่เกินกำหนด
- ใช้ Tranform.translate() เพื่อเคลื่อนย้าย object ที่ player ควบคุม

Vector3.Lerp

Vector3.Lerp เป็น function ที่จะ Return Vector3 ที่จะอยู่ระหว่าง Vector A และ B โดยอิงจากค่า T เป็นหลัก ค่า T คือระหว่าง 0 ถึง 1

รูปแบบของ **Vector3.Lerp** คือ **Vector3.Lerp(Vector3 start, Vector3 finish, float percent)**



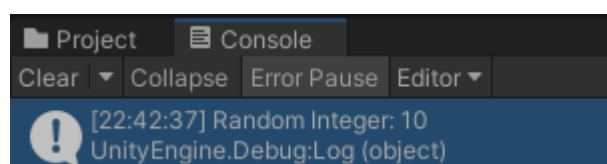
- **start** คือจุดเริ่มต้นของการเปลี่ยนแปลง
- **finish** คือจุดสิ้นสุดของการเปลี่ยนแปลง
- **percent** คือเปอร์เซ็นต์ของระยะทางจาก start ไปยัง end ที่ควรถูกคำนวณ (0 คือ start, 1 คือ end, 0.5 คือจุดกึ่งกลางระหว่าง start และ end)

โดยรวม **Vector3.Lerp** เป็นเครื่องมือที่มีประโยชน์มากในการสร้าง animation และการเคลื่อนไหวที่เรียบร้อยใน Unity.

Random.Range

เป็นคำสั่งที่ใช้ในการสุ่มตัวเลขในช่วงที่เราต้องการได้ โดยคำสั่งนี้ **Random.Range(ค่าเริ่ม, ค่าสิ้นสุด);** โดยจะสุ่มถึงเลขก่อนหน้าค่าสิ้นสุด

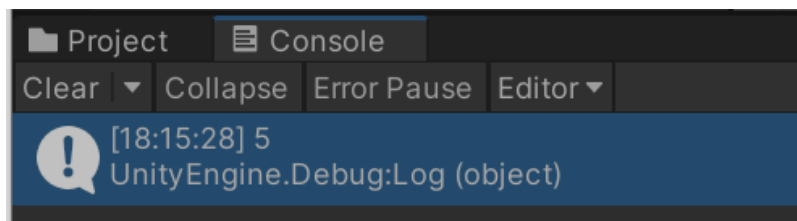
#	Code
1	void Start()
2	{
3	int randomInt = Random.Range(1, 11);
4	Debug.Log("Random Integer: " + randomInt);
5	}



Mathf

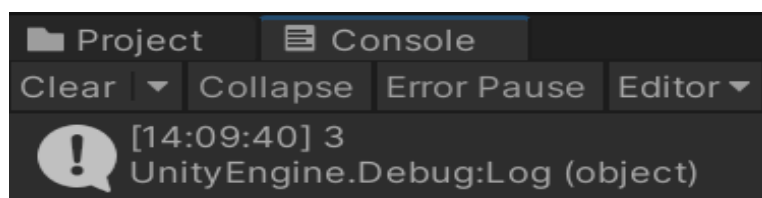
1.Mathf.Abs : ฟังก์ชันนี้ใช้เพื่อคำนวณค่าสัมบูรณ์ (absolute value) ของตัวเลข. ค่าสัมบูรณ์ของตัวเลขคือค่าของตัวเลขโดยไม่สนใจเครื่องหมาย. หมายความว่า **Mathf.Abs(-3)** จะให้ผลลัพธ์เป็น **3** และ **Mathf.Abs(3)** จะให้ผลลัพธ์เป็น **3**.

#	Code
1	void Start()
2	{
3	float myNumber = -5;
4	float absoluteValue = Mathf.Abs(myNumber);
5	Debug.Log(absoluteValue);
6	}



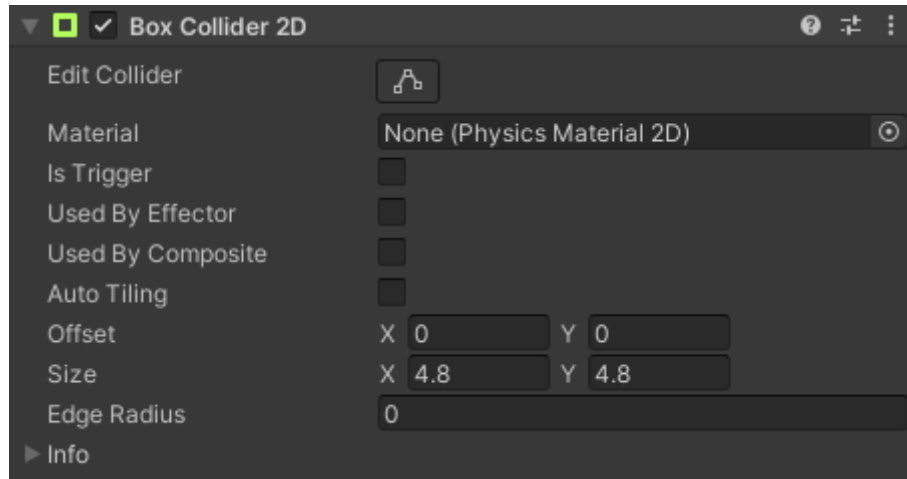
2.Mathf.RoundToInt : ฟังก์ชันนี้ใช้เพื่อปัดค่าทศนิยมของตัวเลขทศนิยมให้กลายเป็นจำนวนเต็มทีใกล้เคียงที่สุด หมายความว่า **Mathf.RoundToInt(2.4f)** จะให้ผลลัพธ์เป็น **2** และ **Mathf.RoundToInt(2.5f)** จะให้ผลลัพธ์เป็น **3**

#	Code
1	void Start()
2	{
3	float myNumber = 2.8f;
4	float roundNumber = Mathf.RoundToInt(myNumber);
5	Debug.Log(roundNumber);
6	}



Collider2D

BoxCollider2D ใช้ในการจำลอง “กรอบพื้นที่” ในการเช็คการชนระหว่าง Object ที่มี Collider2D ด้วยกันเอง (และตัวใดตัวหนึ่งที่ชนจะต้องมี Rigidbody2D อยู่ด้วย) ทำให้เราสามารถตรวจสอบการชนกันของวัตถุผ่านการเขียนโค้ดได้อย่างง่ายดาย



Is Trigger : ปรับชนิดของ Collider ไปเป็น Trigger ซึ่งจะอนุญาตให้ Object นี้สามารถเคลื่อนที่เคลื่อนเข้าไปในกรอบพื้นที่ Collider ของ Object ขึ้นอื่นๆได้ ทำให้ดูเหมือน Object ตัวนี้เคลื่อนที่ทะลุผ่านไป แต่ในความเป็นจริงแล้วเรายังสามารถเช็คการชนได้ของทั้งสองวัตถุได้อยู่

Offset : ใช้การกำหนดตำแหน่งจุดศูนย์กลางของเส้นกรอบที่ใช้ในการตรวจสอบการชน

Size : ใช้ในการกำหนดขนาดของเส้นกรอบได้

Edge Radius : กำหนดค่าความมนของมุม Collider

Rigidbody2D

Rigidbody2Dทำหน้าที่เป็นมวลให้กับObjectชิ้นนั้นเพื่อจำลองสถานการณ์ต่าง ๆ ตามหลักการทางฟิสิกส์ได้แต่ทั้งนี้มวลที่เกิดจากการแปะ Rigidbody2D จะไม่สามารถใช้ตรวจสอบวัตถุชนกันได้ หากไม่มี Collider2D เข้ามาช่วย โดยRigidbody2D มีการตั้งค่าต่าง ๆ ที่ควรรู้ดังนี้

Body type(ชนิดของมวล)

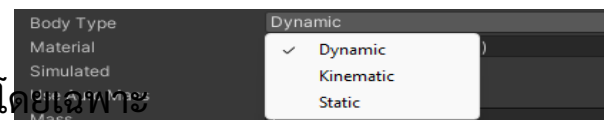
- Dynamic นิยมใช้มากที่สุดเพราะทำให้มวลสามารถเคลื่อนที่ได้ มีการไถลด้วยแรงเสียดทาน , หมุนได้เมื่อชนกับวัตถุอื่น และได้รับผลกระทบจากแรงต่างๆ **เมื่อใช้รูปแบบนี้ไม่ควรใช้งาน**

transform.position และ .rotation

ในการเปลี่ยนตำแหน่งและการหมุนของวัตถุ

- Kinematic ใช้สำหรับควบคุมมวลผ่าน Script โดยเฉพาะ และไม่ได้รับผลกระทบจากแรงต่าง ๆ อีกทั้งยังเช็คการชนได้กับเฉพาะมวลแบบ Dynamic เท่านั้น

- Static ใช้กับมวลที่ต้องการให้มันหยุดอยู่นิ่งตลอดเวลา ไม่ได้รับผลจากแรงต่าง ๆ และสามารถเช็คการชนได้กับมวลแบบ Dynamic เท่านั้น



การตั้งค่า	คำอธิบาย
Simulated	ถ้าไม่เลือกตัวเลือกนี้จะไม่มีการจำลองมวลเกิดขึ้นรวมถึงหยุดเช็คการชนกับวัตถุอื่นด้วย
Use AutoMass	เลือกตัวเลือกนี้เพื่อให้มีการคำนวณมวลอัตโนมัติจากไซส์ของ Collider
Mass	ใช้กำหนดมวลให้กับวัตถุ ยิ่งมวลมากการออกแรงให้มวลนี้เคลื่อนที่หรือหยุดนิ่งจะยิ่งต้องอาศัยแรงมากขึ้น

การตั้งค่า	คำอธิบาย
Linear Drag	แรงเสียดทานเชิงเส้นยิ่งมากวัตถุจะยิ่งเคลื่อนที่ฝืดเมื่อเสียดสีกับพื้นผิวของมวลอื่น
Angular Drag	แรงเสียดทานเชิงมุมยิ่งมากวัตถุจะหมุนไปมาได้ยากขึ้น เมื่อมีแรงตกกระทบที่มุมของวัตถุ
Gravity Scale	แรงโน้มถ่วงที่กระทำต่อวัตถุนั้นๆ ถ้าไม่ใช่เป็น 0 วัตถุจะร่วงลงไปตามแนวแกน y
Collision Detection	กำหนดวิธีใช้การชน
Discrete	ทำให้บางครั้งหาก Object นี้เคลื่อนที่เร็วเกินกำหนดจะเคลื่อนทะลุผ่านมวลที่ชนไปได้
Continuous	ทำให้ Object ไม่สามารถทะลุผ่านมวลที่ชนไปได้เลย
Constraints	กำหนดข้อจำกัดด้านการเคลื่อนไหวให้กับ Object
Freeze Position	ล็อกไม่ให้ Object นั้นเคลื่อนไหวตามแนว X, Y, และ Z
Freeze Rotation	ล็อกไม่ให้ Object เกิดการหมุน

Ontrigger2D & OnCollision2D

Collision ต่างกับ trigger ยังไง ?

Collision : เมื่อชนกัน object จะเคลื่อนที่ไปตามแรงของฟิสิกส์

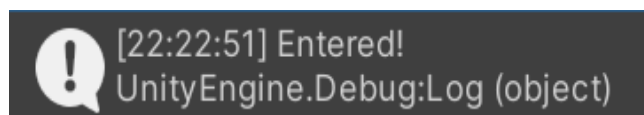
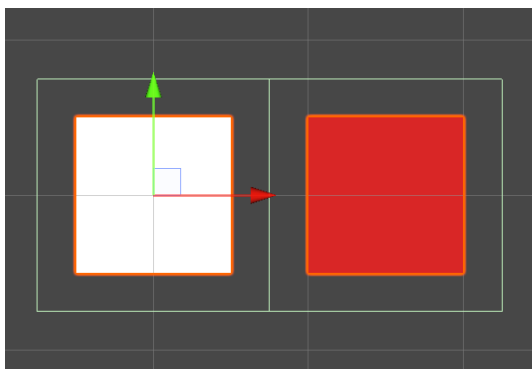
Trigger : เมื่อชนกันวัตถุจะทะลุไปเลยซ้อนทับกันได้ จะเกิดขึ้นเมื่อเราใช้
Istrigger

- Stay: ทำตลอดเวลาที่สัมผัสกัน
- Enter: จะทำงานแค่ 1 ครั้งเมื่อชน
- Exit: ทำงานเมื่อวัตถุแยกออกจากกัน

ตัวอย่างคำสั่งในการใช้งาน

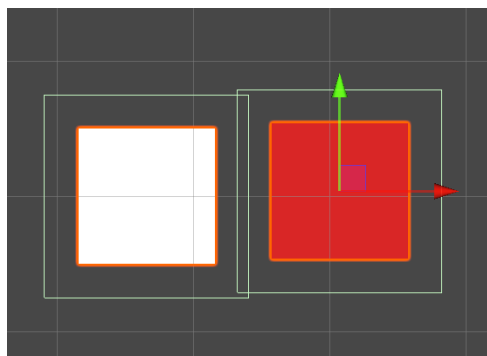
OnTriggerEnter2D จะทำโค้ดในฟังก์ชันนี้เมื่อมีวัตถุมาชนมัน (ชน 1 ครั้งจะทำ 1 ครั้ง)

#	Code
1	void OnTriggerEnter2D(Collider2D other)
2	{
3	if (other.CompareTag("Enemy"))
4	{
5	Debug.Log("Entered!");
6	}
7	}



OnTriggerStay2D ทำตลอดเวลาที่สัมผัสกัน

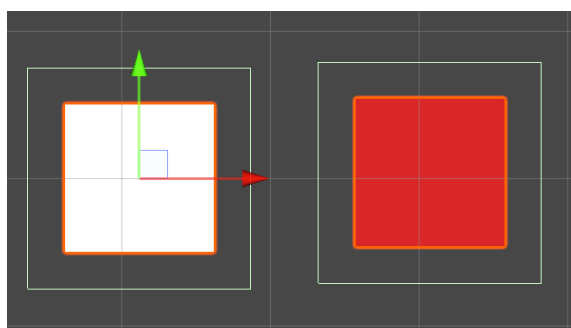
#	Code
1	void OnTriggerStay2D(Collider2D other)
2	{
3	if (other.CompareTag("Enemy"))
4	{
5	Debug.Log("Staying!");
6	}
7	}



[22:22:51] Staying!
UnityEngine.Debug:Log (object)

OnTriggerExit2D ทำเมื่อวัตถุแยกจากกันหลักสัมผัสกันเรียบร้อยแล้ว

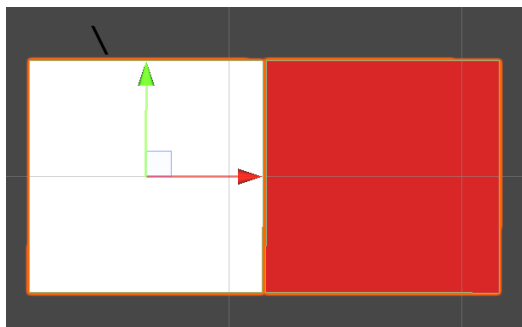
#	Code
1	void OnTriggerExit2D(Collider2D other)
2	{
3	if (other.CompareTag("Enemy"))
4	{
5	Debug.Log("Exit!");
6	}
7	}



[22:23:00] Exited!
UnityEngine.Debug:Log (object)

OnCollisionEnter2D จะทำได้ในฟังก์ชันนี้เมื่อมีวัตถุมาชนมัน (ชน 1 ครั้งจะทำ 1 ครั้ง) แต่วัตถุจะถูกดันออกเล็กน้อย

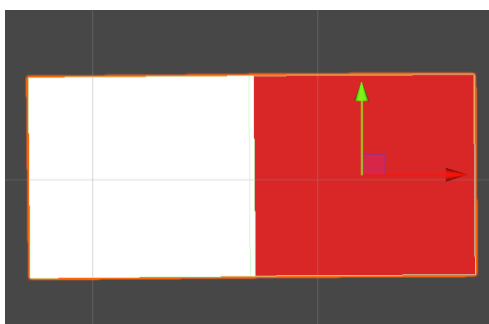
#	Code
1	void OnCollisionEnter2D(Collision2D collision)
2	{
3	if(collision.gameObject.CompareTag("Enemy"))
4	{
5	Debug.Log("Entered!");
6	}
7	}



[23:27:15] Entered!
UnityEngine.Debug.Log (object)

OnCollisionStay2D ทำตลอดเวลาที่สัมผัสกัน

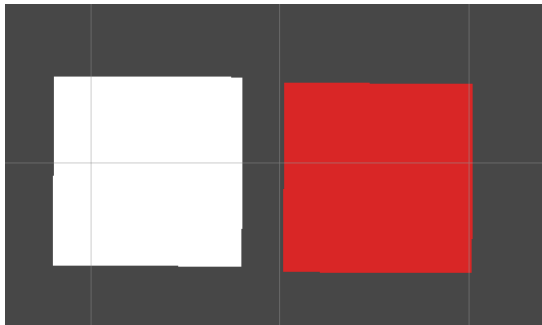
#	Code
1	void OnCollisionStay2D(Collision2D collision)
2	{
3	if(collision.gameObject.CompareTag("Enemy"))
4	{
5	Debug.Log("Staying!");
6	}
7	}



[23:51:46] Staying!
UnityEngine.Debug.Log (object)


OnCollisionExit2D ทำเมื่อวัตถุแยกจากกันหลักสัมผัสกันเรียบร้อยแล้ว

#	Code
1	<code>void OnCollisionExit2D(Collision2D collision)</code>
2	<code>{</code>
3	<code> if(collision.gameObject.CompareTag("Enemy"))</code>
4	<code> {</code>
5	<code> Debug.Log("Exit!");</code>
6	<code> }</code>
7	<code>}</code>



[00:05:50] Exited!
UnityEngine.Debug:Log (object)

gameObject.tag

ใน Inspector  Inspector เราใช้ Tag Enemy ในการกำหนดว่า Object นี้คือชนิดไหนเช่น Player หรือ Enemy เราใช้การกำหนดใน Script ว่าเมื่อเกิดการชนกันและ Object ที่ชนมี Tag อะไรจึงจะทำงานแบบไหน เช่นเมื่อชนกับ Tag Player ก็อาจจะไม่มีอะไรเกิดขึ้นแต่หากชนกับ Tag Enemy ก็ทำให้เกิดการทำงานอย่างหนึ่งขึ้นมาได้ตาม Script ที่ได้เขียน

#	Code
1	void OnTriggerEnter2D(Collider2D collision)
2	{
3	if (collision.gameObject.tag == "Enemy")
4	{
5	Debug.Log("Exit!");
6	}
7	}

Destroy()

คำสั่งสำหรับทำลาย Object ให้หายไปจาก Scene

โดยเราสามารถกำหนดเวลาดีเลย์ได้ด้วยนะ

รูปแบบการใช้งาน : Destroy (Object obj, float t); อธิบาย obj

สิ่งที่ต้องการจะทำลายทิ้ง เช่น ถ้าต้องการทำลายตัวเองให้ใส่ this.gameObjectt กำหนดเวลาดีเลย์ ก่อนจะทำลายเป้าหมาย

#	Code
1	void Start()
2	{
3	Destroy(this.gameObject , 10f);
4	}

Prefab

ระบบ Prefab คือระบบที่เราสามารถแก้ไข Object ใน Scene ที่หน้าตาเหมือนกันด้วยการแก้ไขเพียงครั้งเดียวยกตัวอย่างง่าย ๆ เช่น ใน Scene ของเราวางต้นไม้ไว้ 100 ต้นซึ่งเป็นต้นไม้สีเขียว แล้วจู่ ๆ เราต้องการเปลี่ยนสีของต้นไม้ทั้ง 100 ต้นใน Scene ให้กลายเป็นต้นไม้สีแดงแทนเราสามารถทำ Object ต้นไม้ให้เป็น Prefab ไว้ก่อนได้แล้วค่อยก๊อปปี้นออกมาวางเยอะๆใน Scene ในอนาคตถ้าเราต้องการเปลี่ยนลักษณะสีของมัน ก็สามารถทำได้โดยการแก้ไขที่ตัว Prefab ต้นแบบเพียงครั้งเดียว แล้ว Unity จะทำการอัปเดตสีของต้นไม้ต้นอื่นๆที่สร้างมาจาก Prefab นั้นทำให้เราไม่ต้องมาไล่แก้ทั้ง 100 ต้นนั่นเอง

- การสร้าง Prefab ทำได้ง่ายมาก ๆ แค่ลาก Object ใน **hierarchy** ที่ต้องการให้เป็น Prefab เข้าไปปล่อยในหน้าต่าง **Project** ก็เป็นอันเรียบร้อยจ้า
- การ แก้ไขต่าง ๆ บน Prefab จะไม่ถูกเซฟจนกว่าจะกด Apply ที่หน้าต่างInspector ของ Prefab นั้น

Instantiate

คำสั่งสำหรับคัดลอก (Copy) Object ที่เราต้องการขึ้นมาเพิ่มใน Scene ได้ หรือจะคัดลอก Prefab ต่าง ๆ ใน Library มาปรากฏใน Scene ก็ทำได้เช่นกัน โดยถ้า Object หรือ Prefab นั้นมี Child ตัว Copy ก็จะถูกก๊อปออกมาให้มี Child

ด้วยรูปแบบการใช้งาน :Instantiate(Object original, Vector3 position, Quaternion rotation); อธิบาย

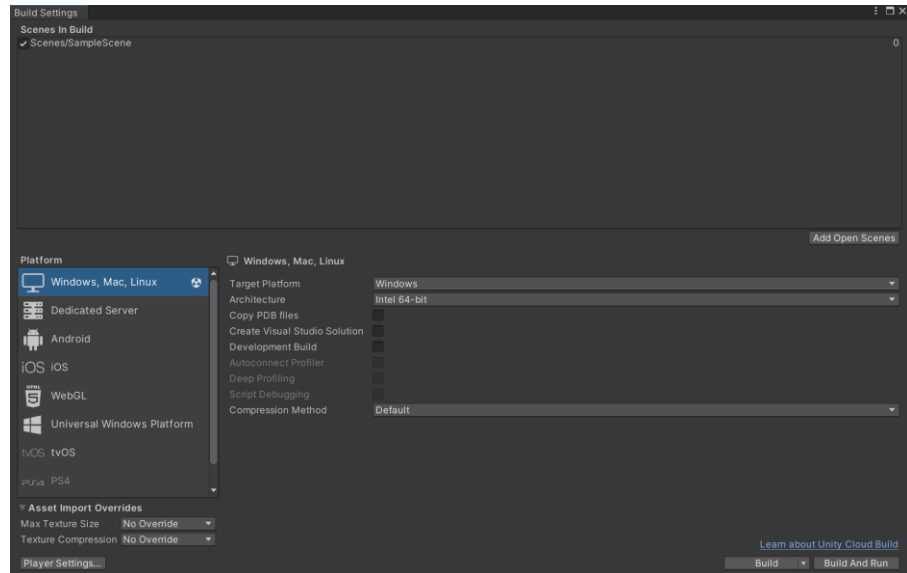
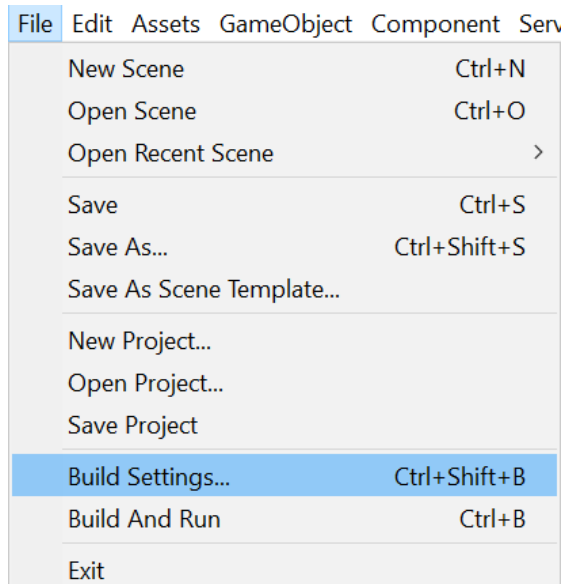
original วัตถุดั้งเดิม (ตัวต้นแบบ) ที่เราต้องการ Copy ซึ่งส่วนใหญ่เป็นตัวต้นแบบ

position ตำแหน่งที่ต้องการให้ตัว Copy ของเราปรากฏขึ้นเป็น Vector3

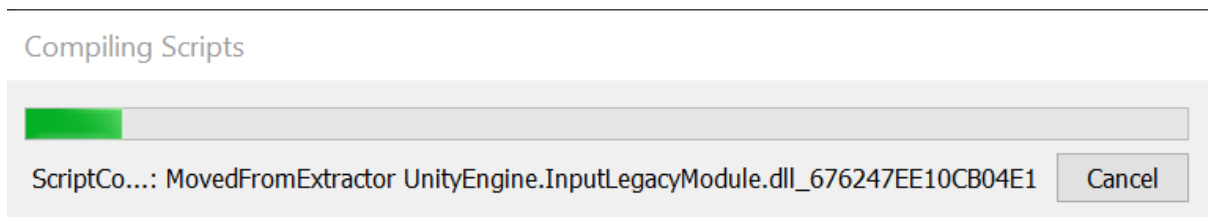
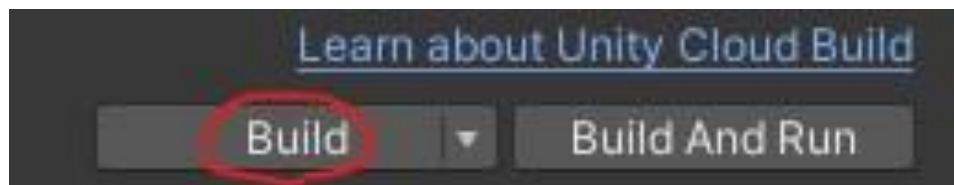
rotation การหมุนมุมของตัว Copy ที่เราสร้างขึ้นมาปกติเราจะใส่เป็น Quaternion.identity

การBuildเกมใน Unity

1.คลิกที่ File -> Build Settings หรือสามารถกดคีย์ลัดคือ **Ctrl + Shift + B** และลาก Scene ที่เราต้องการจะบีวจากแถบ Project ด้านล่างมาที่ Scenes in Bulid



2.กดปุ่ม Build เพื่อให้เกมเป็นไฟล์ .exe แล้วรอนจนเสร็จ





I-DIA CAMP X