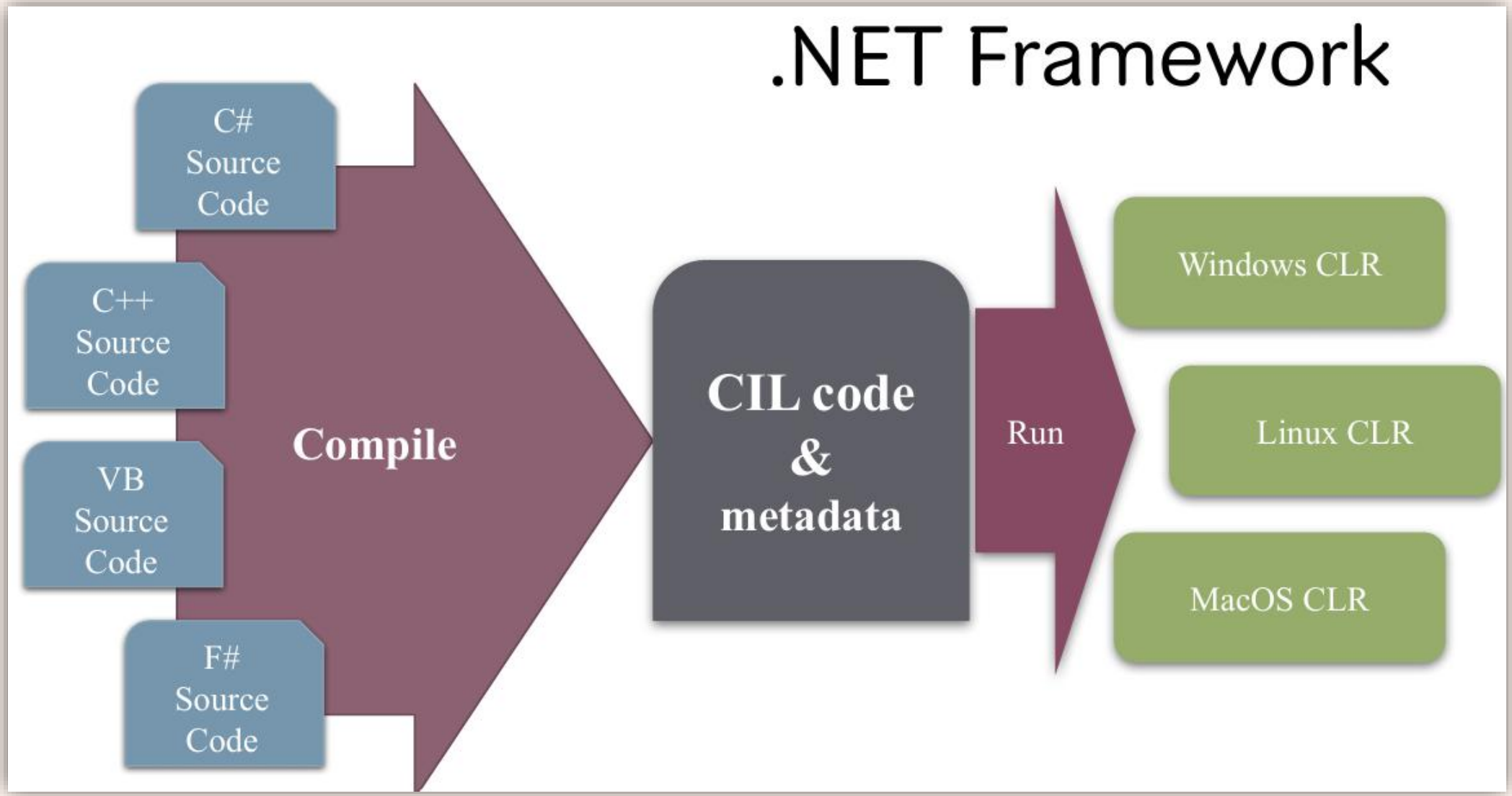
วันที่ 20 มิถุนายน 2569

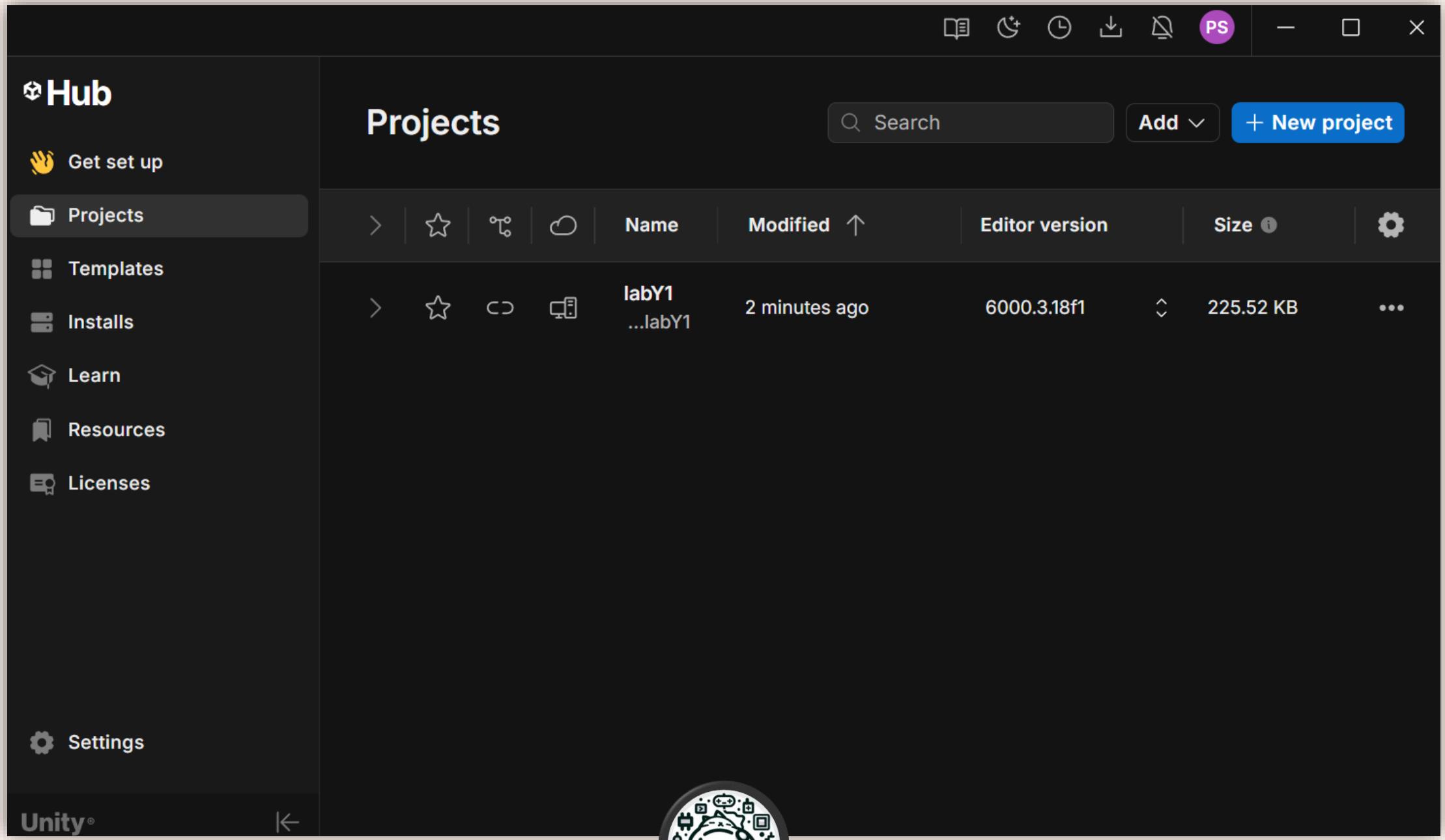
อบรมเชิงปฏิบัติการเรื่อง C#

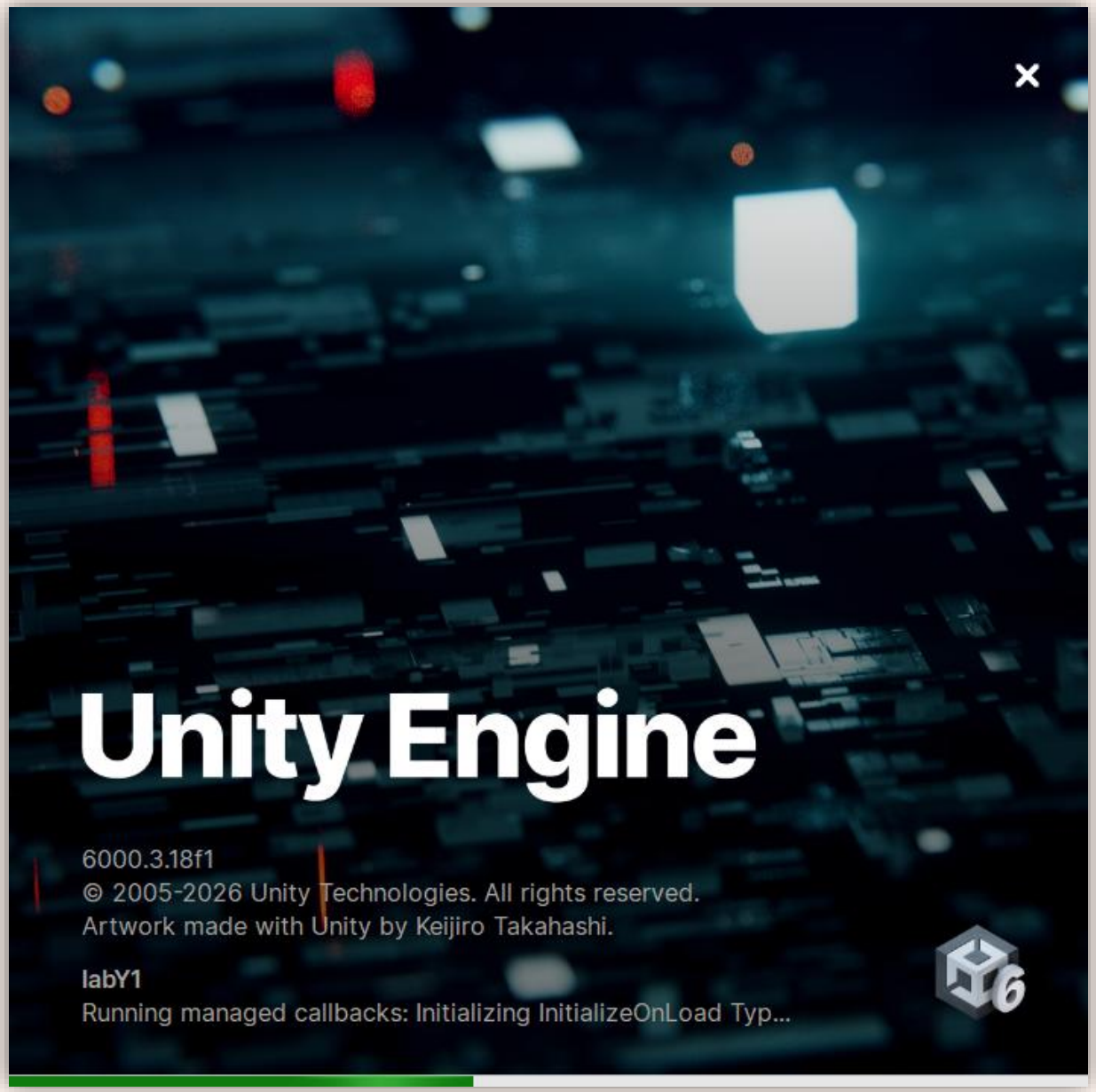
สำหรับพัฒนาเกมเพื่อการเรียนรูตลอดชีวิต Life long Learning

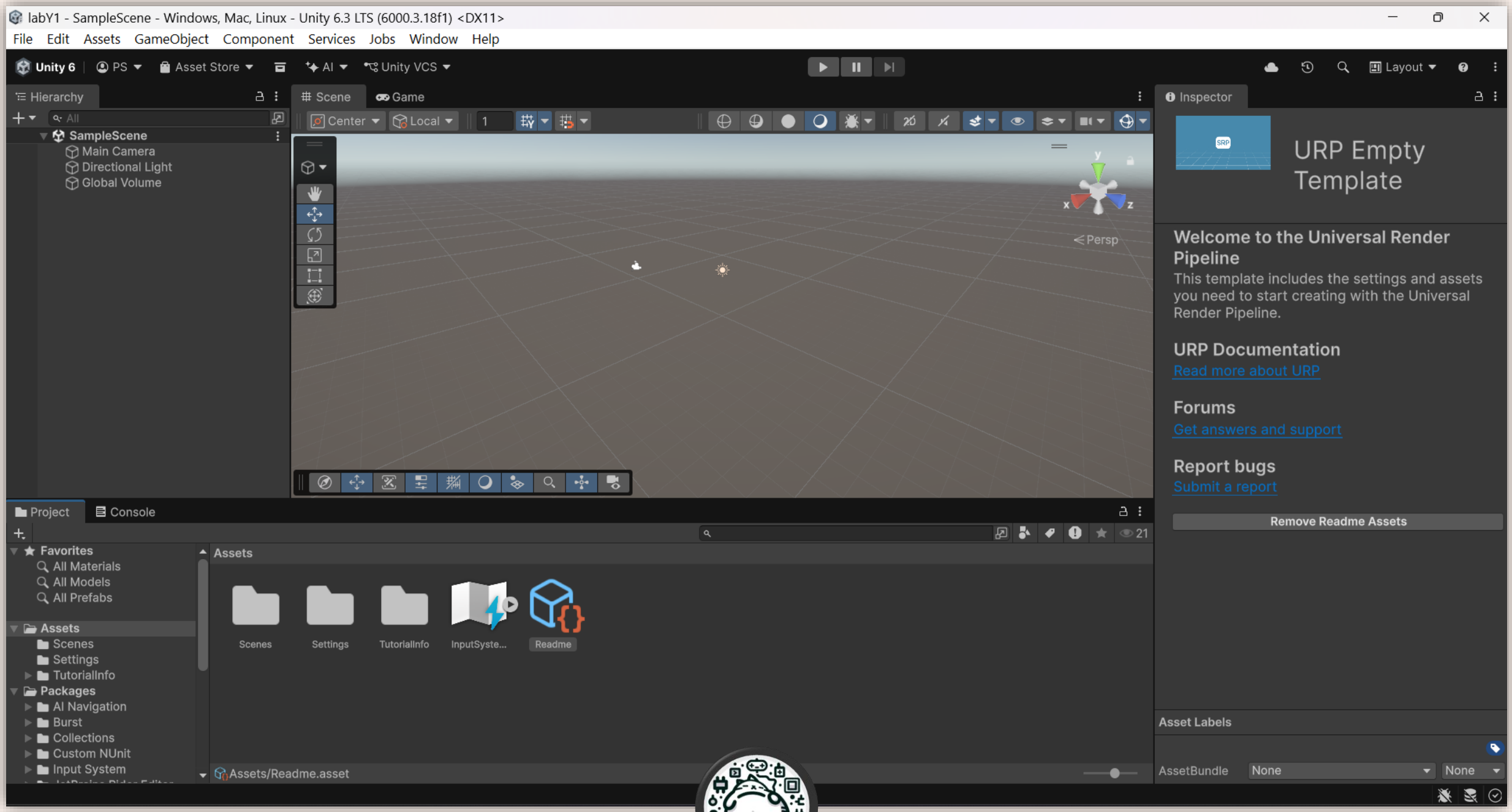
อาจารย์ดนัย เจษฎาฐิติกุล
ผู้ช่วยศาสตราจารย์จารุต บุศราทิจ
ดิจิทัลคอนเทนต์และเกม@PBRU

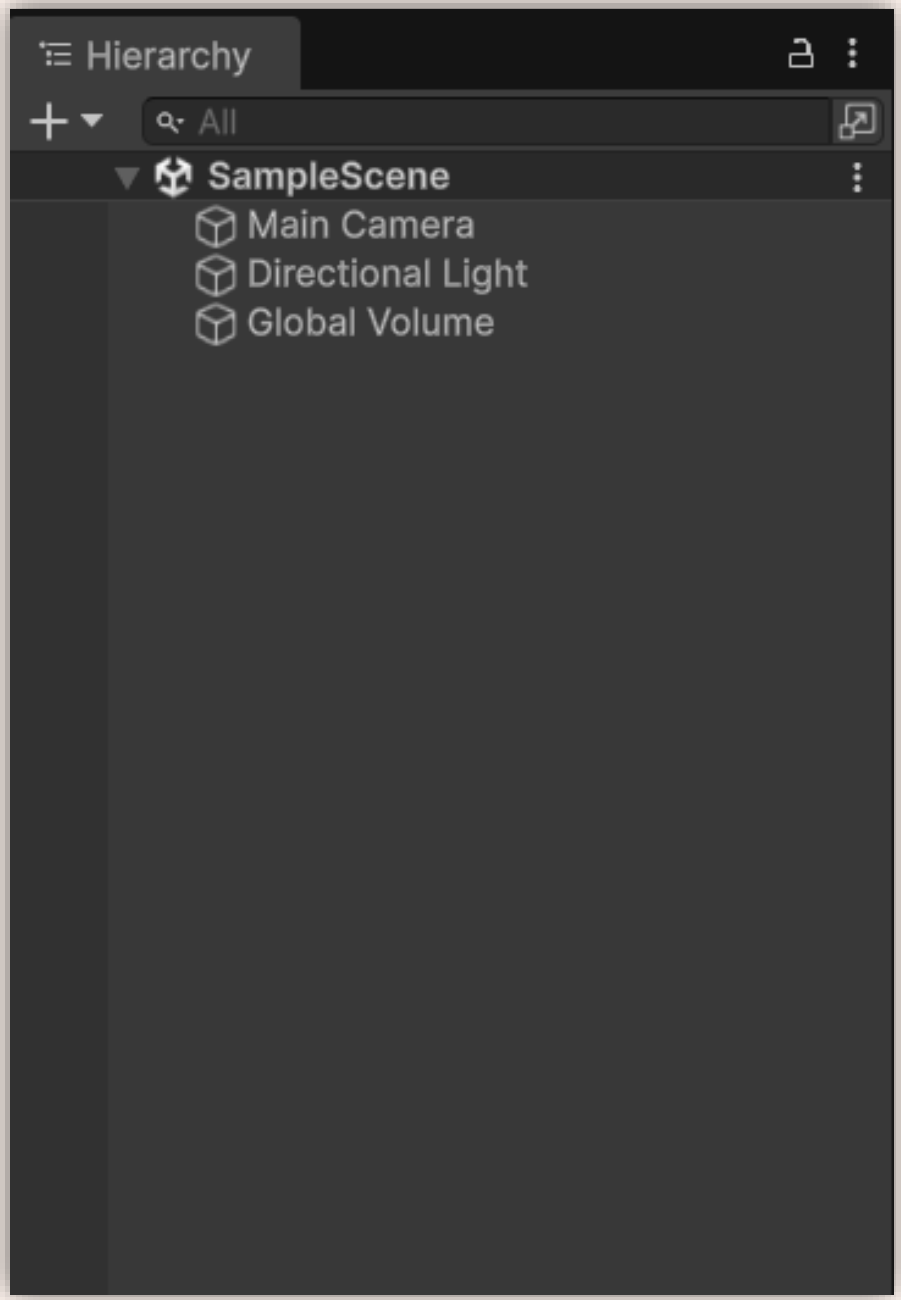

.NET Framework
C# Source Code
C++ Source Code
VB Source Code
F# Source Code
Compile
CIL code & metadata
Run
Windows CLR
Linux CLR
MacOS CLR

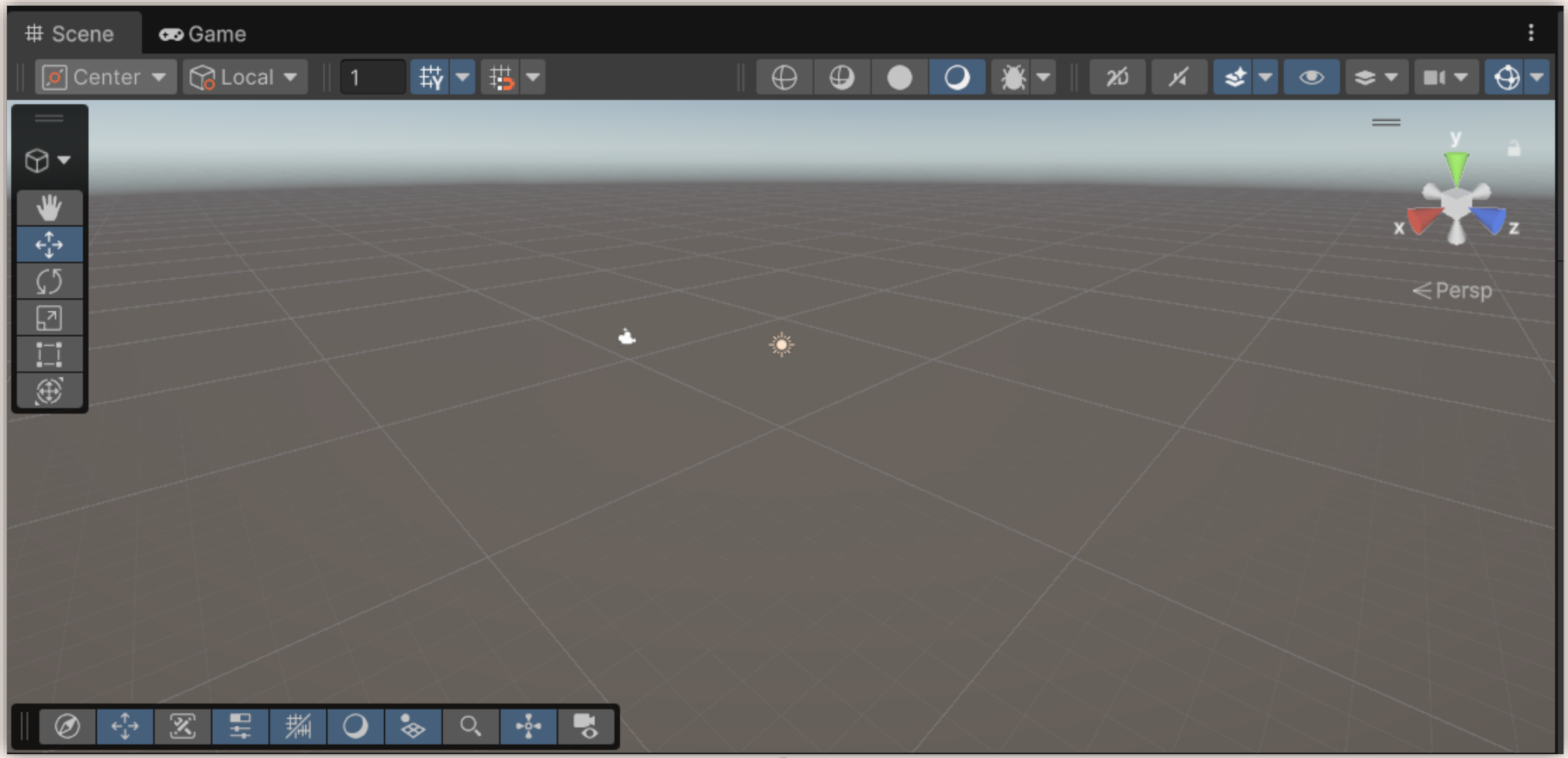
แนวคิด Unity & C# Structure

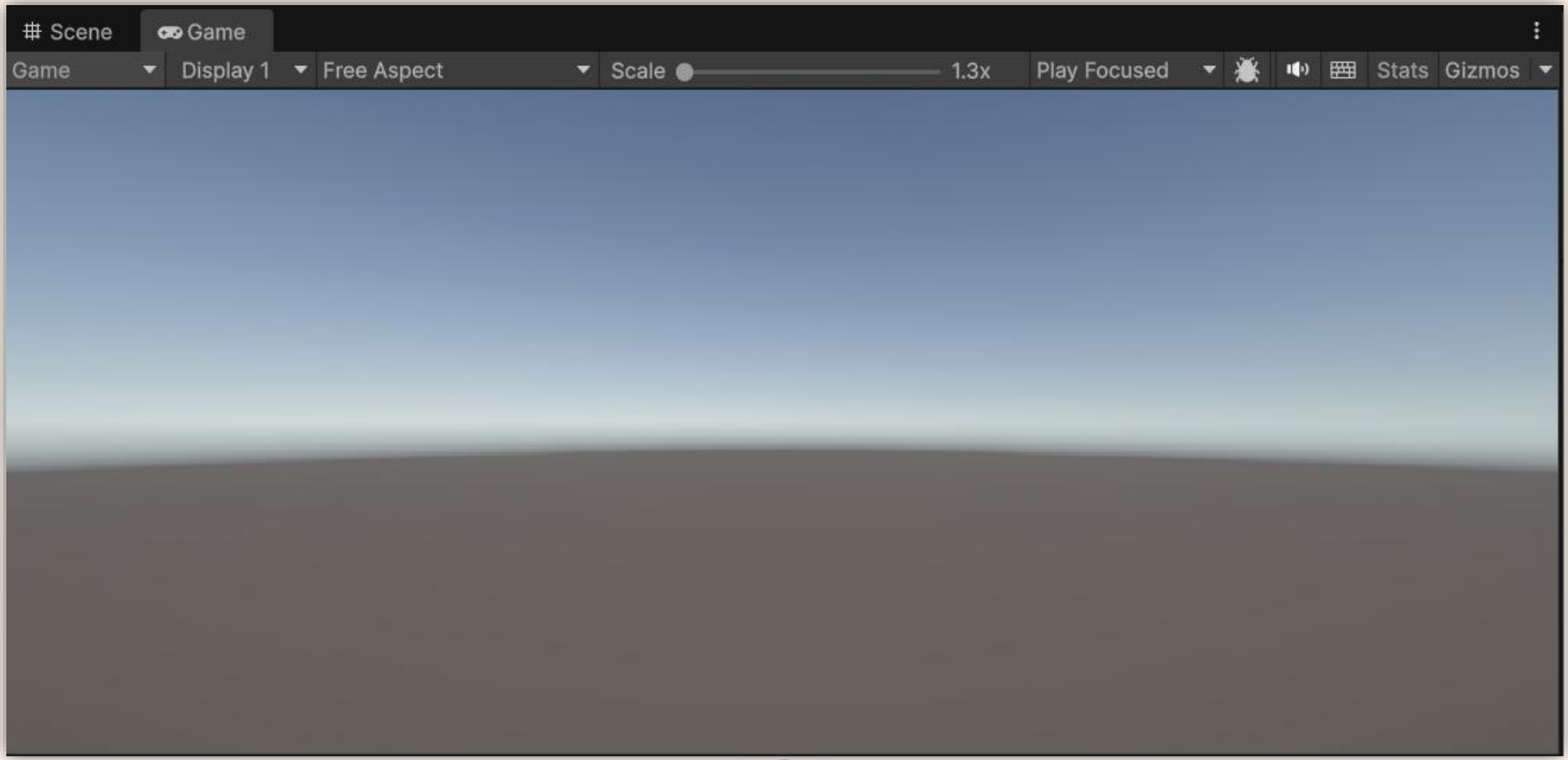

Hub
Get set up
Projects
Templates
Installs
Learn
Resources
Licenses
Settings
Unity
Projects
Search
Add
New project
Name
Modified
Editor version
Size
labY1
...labY1
2 minutes ago
6000.3.18f1
225.52 KB

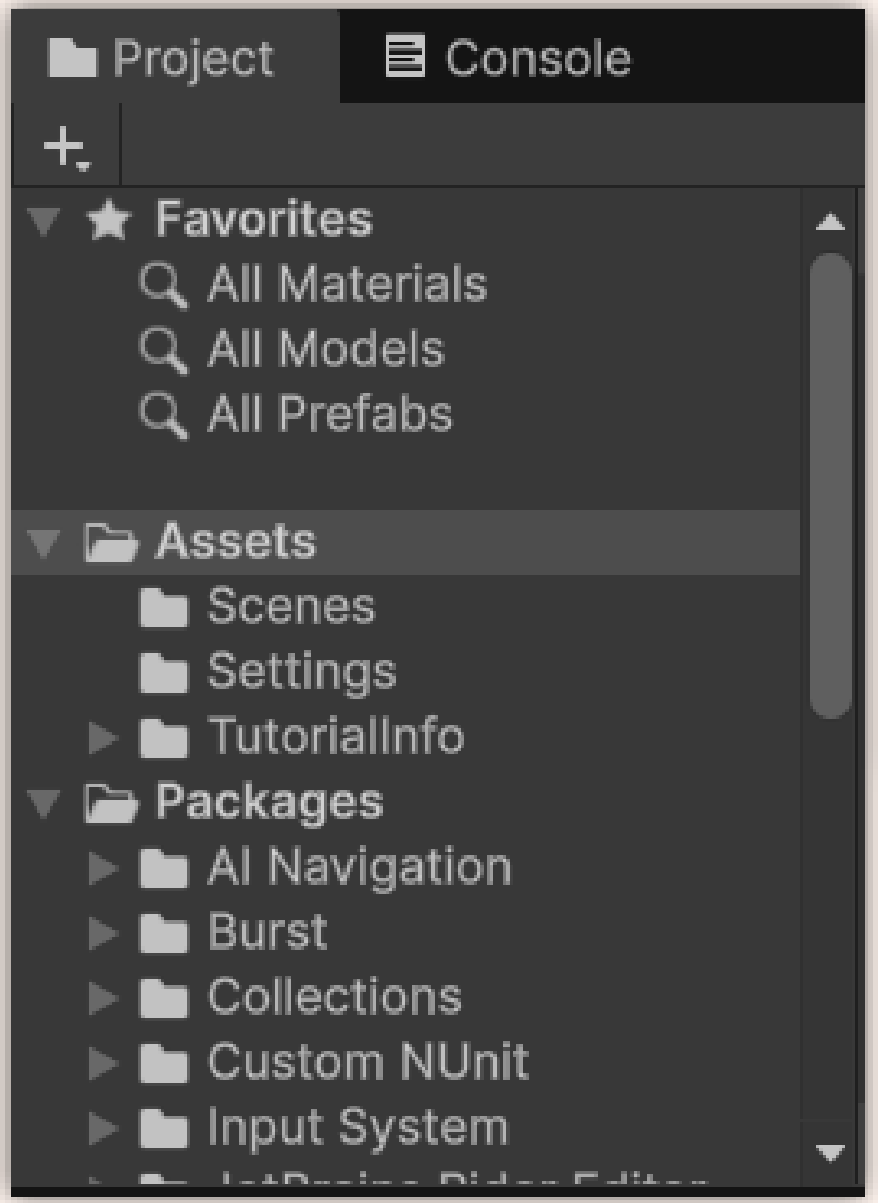

Unity Engine
6000.3.18f1
© 2005-2026 Unity Technologies. All rights reserved.
Artwork made with Unity by Keijiro Takahashi.
labY1
Running managed callbacks: Initializing InitializeOnLoad Typ...

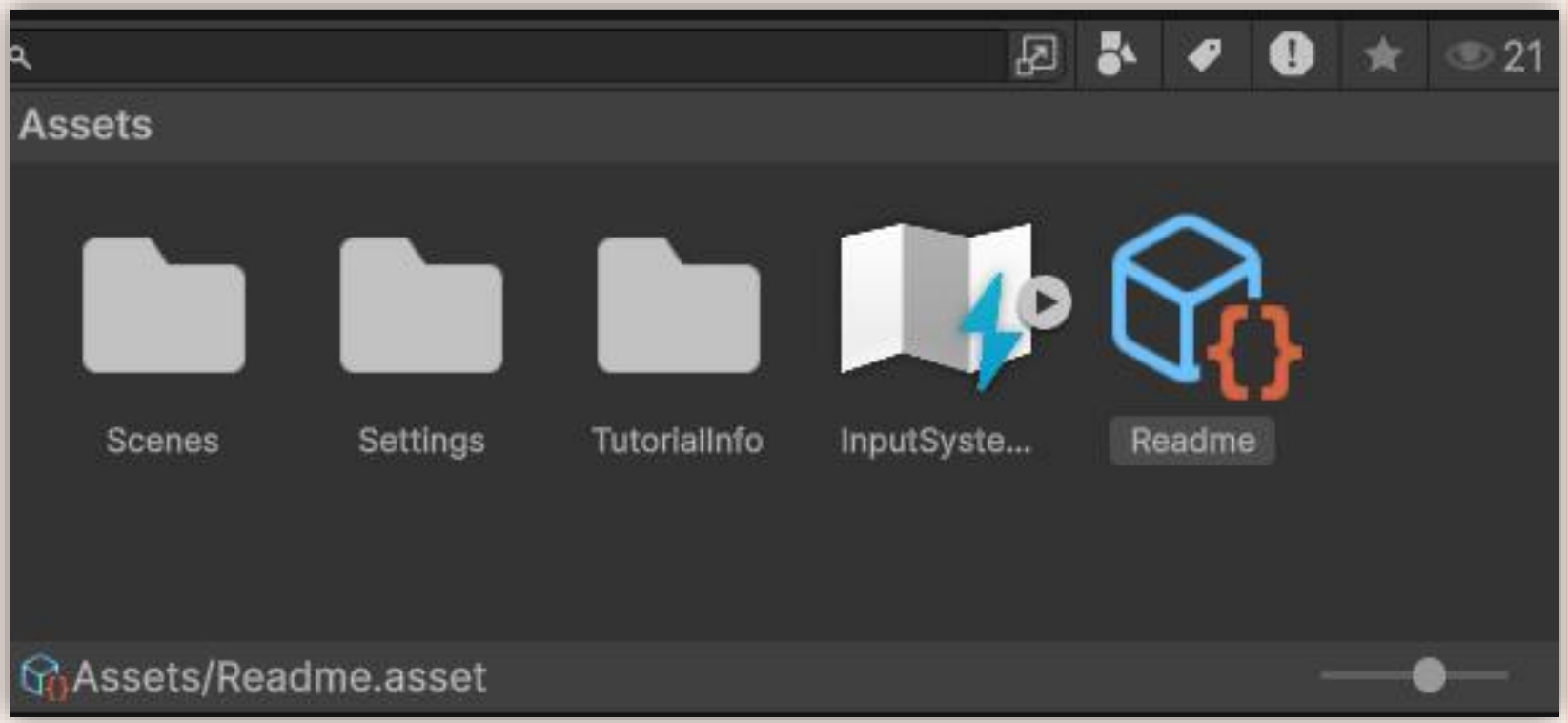

labY1 - SampleScene - Windows, Mac, Linux - Unity 6.3 LTS (6000.3.18f1) <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
Hierarchy
SampleScene
Main Camera
Directional Light
Global Volume
Scene
Game
Persp
Inspector
URP Empty Template
Welcome to the Universal Render Pipeline
This template includes the settings and assets you need to start creating with the Universal Render Pipeline.
URP Documentation
Read more about URP
Forums
Get answers and support
Report bugs
Submit a report
Remove Readme Assets
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Custom NUnit
Input System
Scenes
Settings
TutorialInfo
InputSyste...
Readme
Assets/Readme.asset
Asset Labels
AssetBundle
None
None

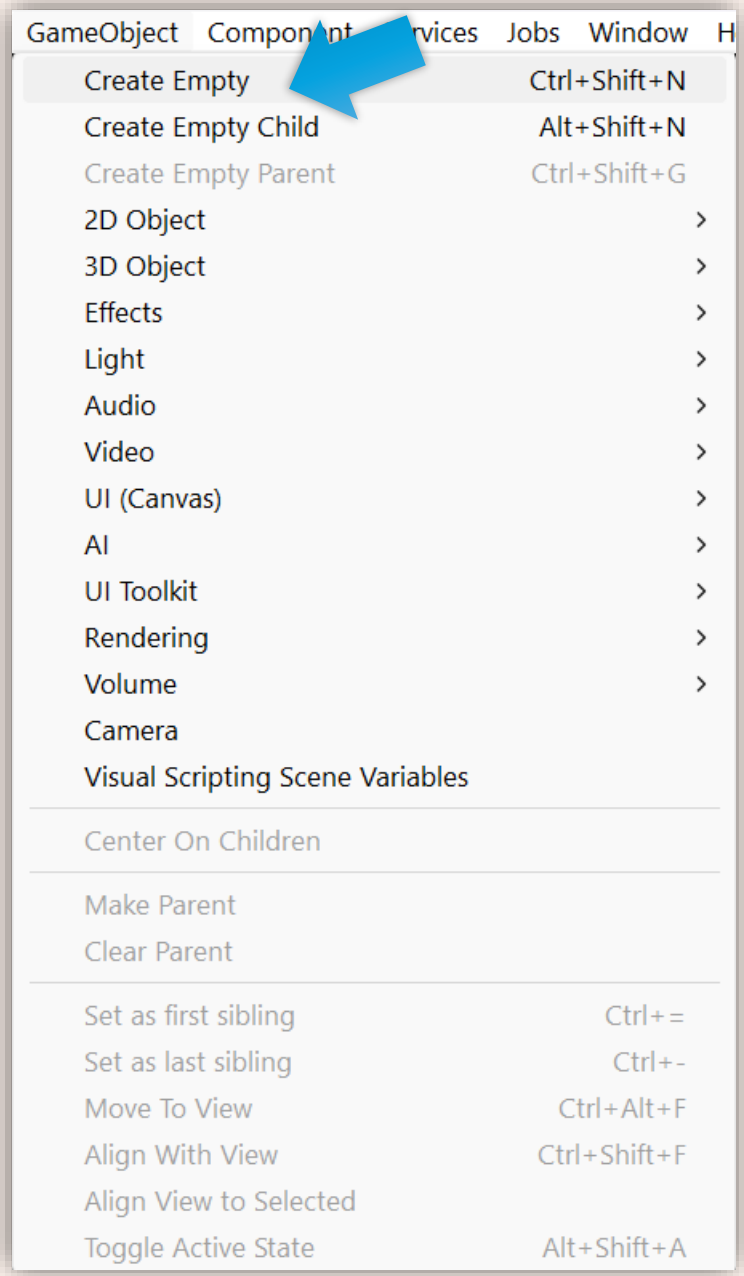

Hierarchy
All
SampleScene
Main Camera
Directional Light
Global Volume

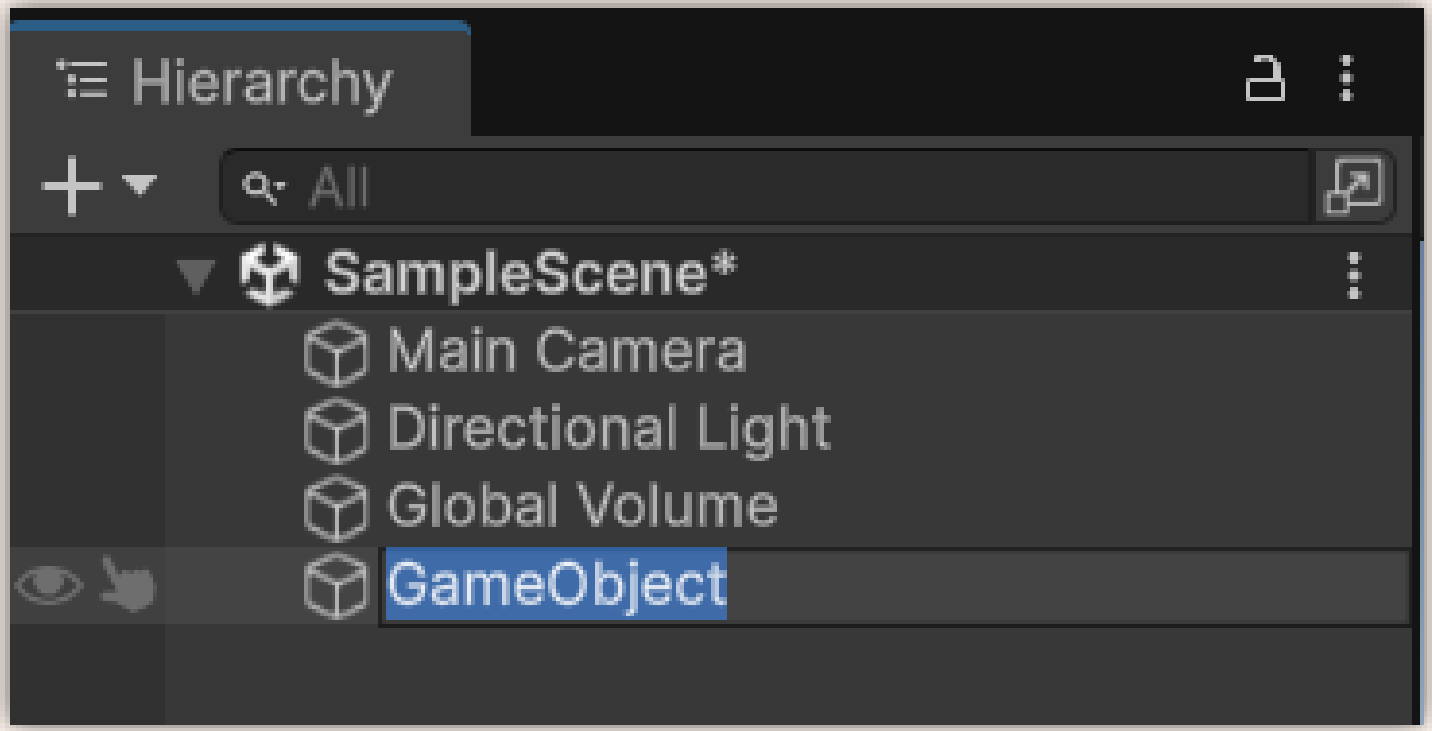

Scene
Game
Center
Local
1
y
x
z
Persp

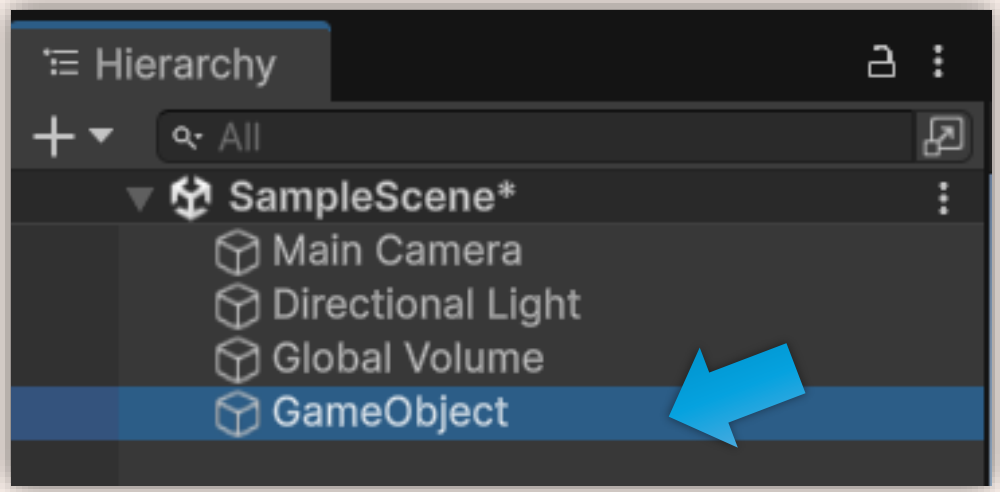

Scene
Game
Game
Display 1
Free Aspect
Scale
1.3x
Play Focused
Stats
Gizmos

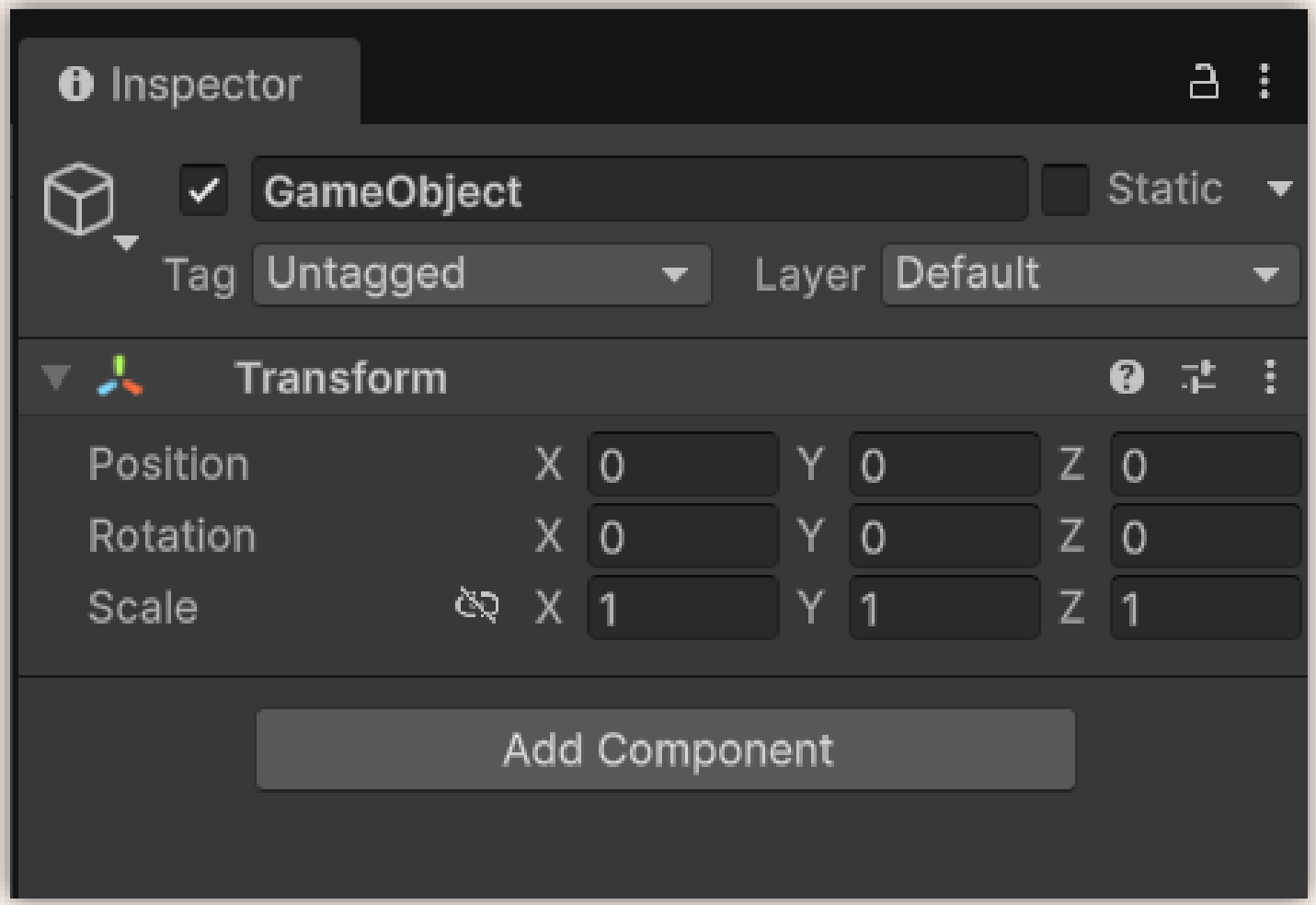

Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Settings
TutorialInfo
Packages
AI Navigation
Burst
Collections
Custom NUnit
Input System

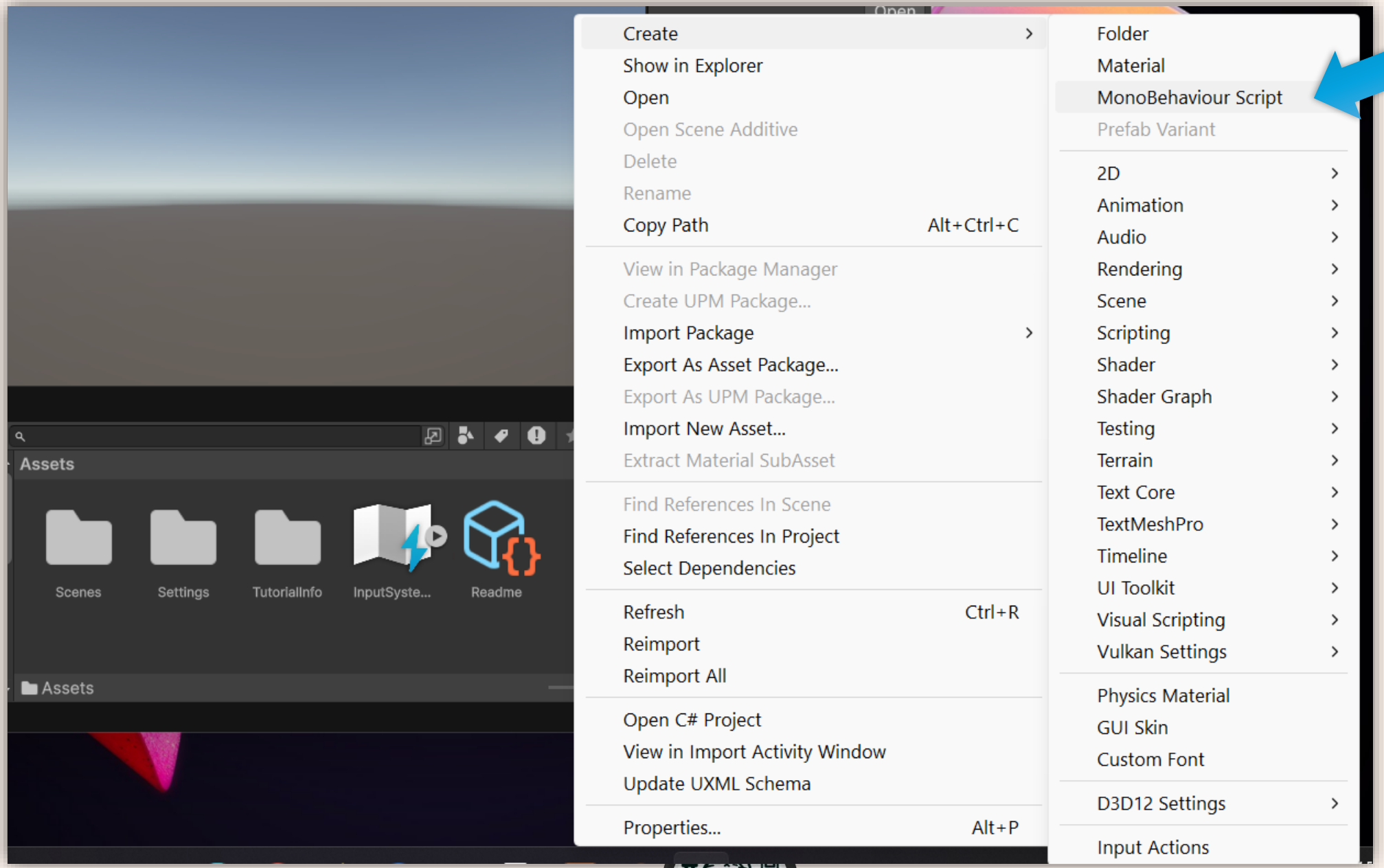

Assets
Scenes
Settings
TutorialInfo
InputSyste...
Readme
Assets/Readme.asset

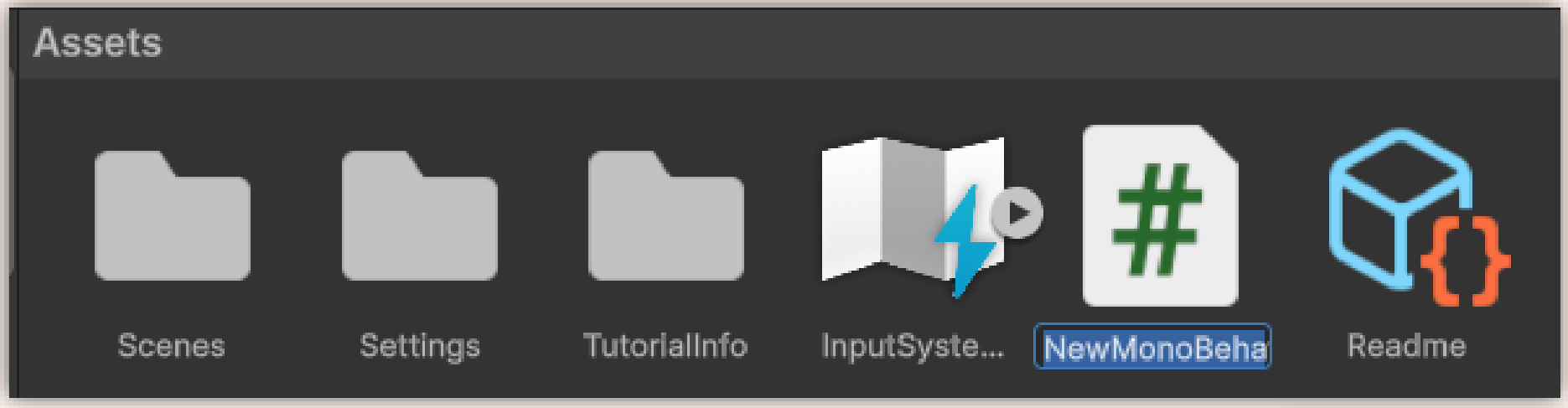

GameObject
Jobs
Window
Create Empty Ctrl+Shift+N
Create Empty Child Alt+Shift+N
Create Empty Parent Ctrl+Shift+G
2D Object
3D Object
Effects
Light
Audio
Video
UI (Canvas)
AI
UI Toolkit
Rendering
Volume
Camera
Visual Scripting Scene Variables
Center On Children
Make Parent
Clear Parent
Set as first sibling Ctrl+=
Set as last sibling Ctrl+-
Move To View Ctrl+Alt+F
Align With View Ctrl+Shift+F
Align View to Selected
Toggle Active State Alt+Shift+A


Hierarchy
All
SampleScene*
Main Camera
Directional Light
Global Volume
GameObject

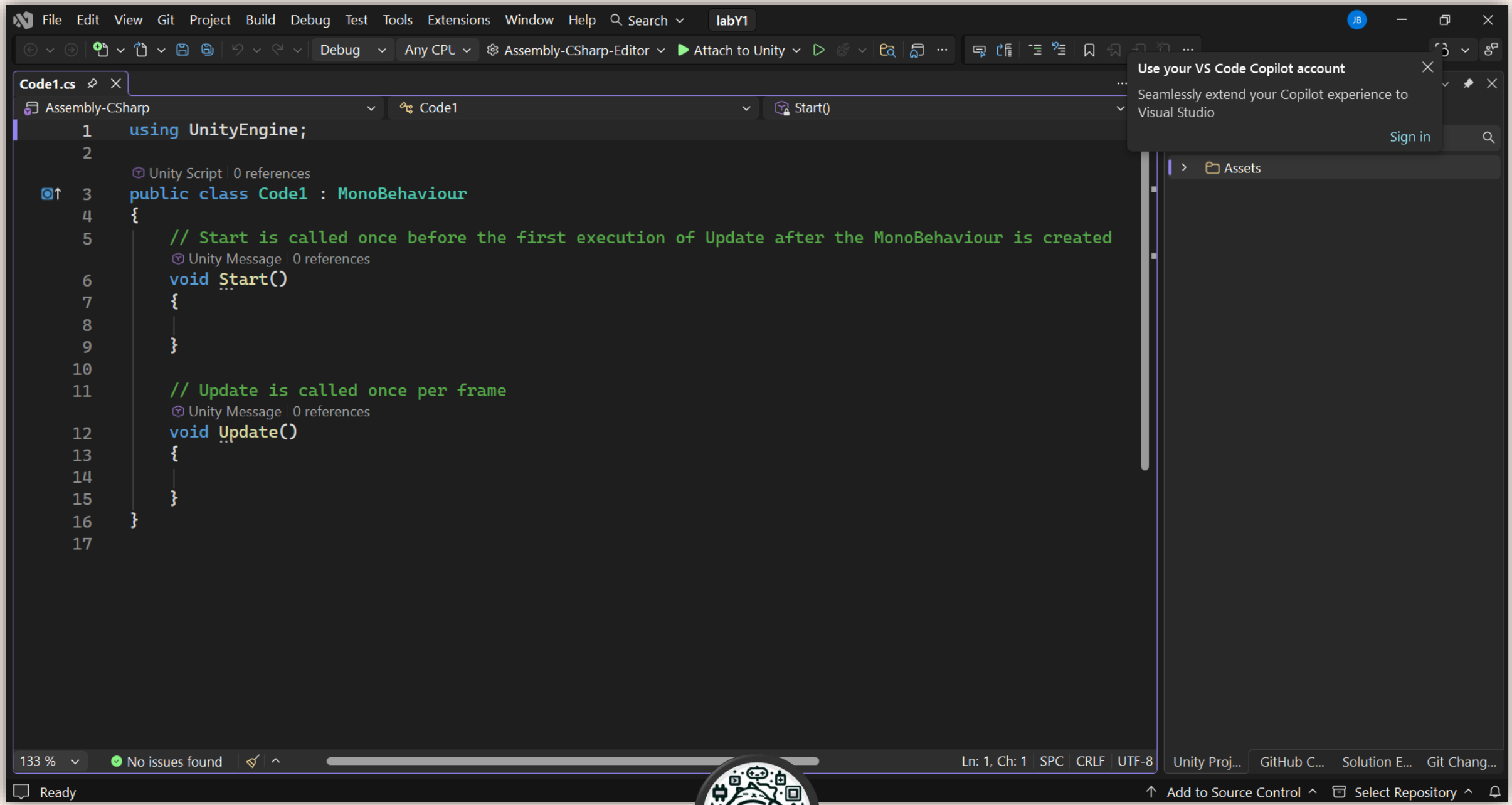

Hierarchy
All
SampleScene*
Main Camera
Directional Light
Global Volume
GameObject

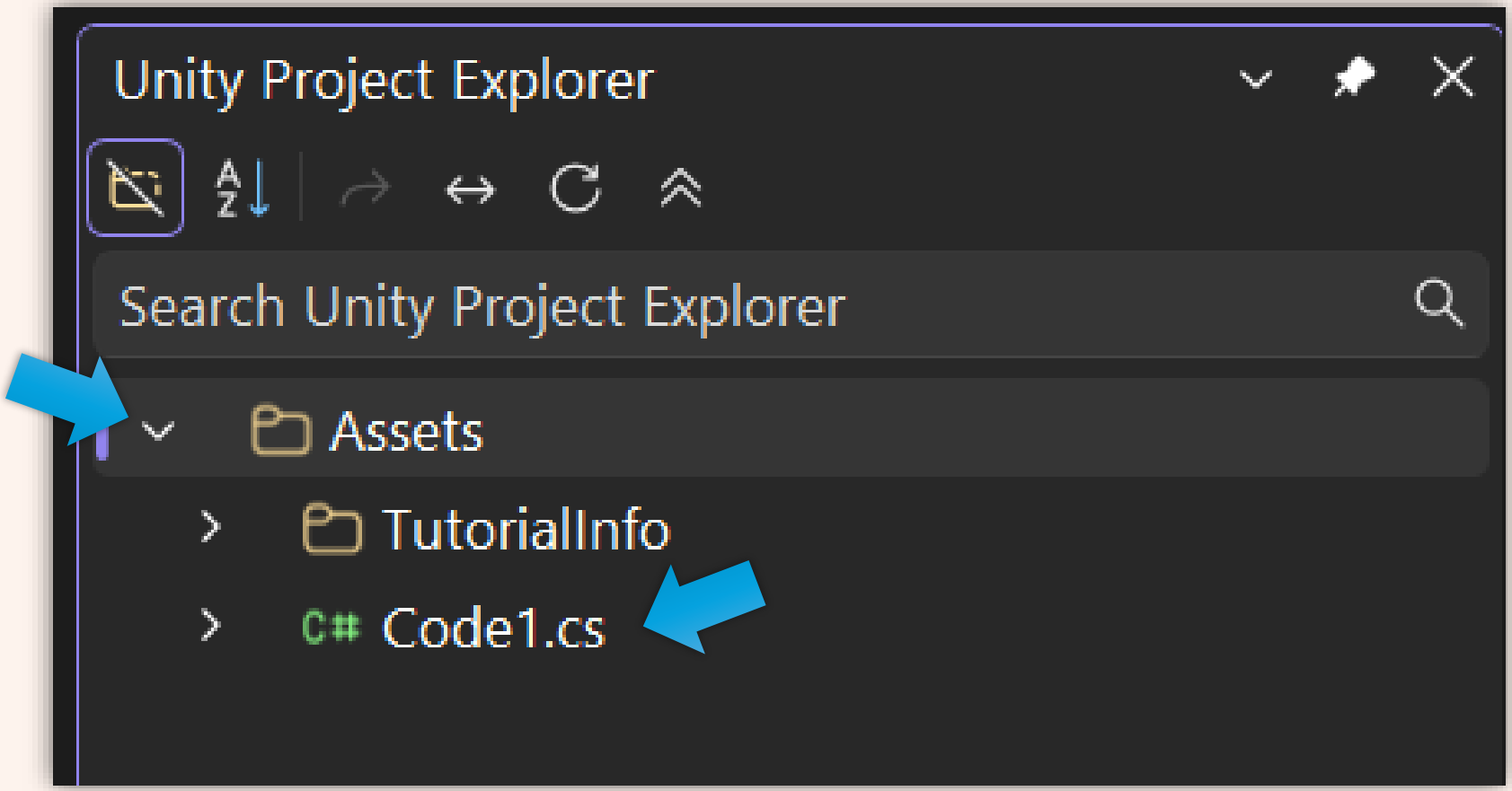

Inspector
GameObject
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Add Component


Create
Show in Explorer
Open
Open Scene Additive
Delete
Rename
Copy Path Alt+Ctrl+C
View in Package Manager
Create UPM Package...
Import Package
Export As Asset Package...
Export As UPM Package...
Import New Asset...
Extract Material SubAsset
Find References In Scene
Find References In Project
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Open C# Project
View in Import Activity Window
Update UXML Schema
Properties... Alt+P
Folder
Material
MonoBehaviour Script
Prefab Variant
2D
Animation
Audio
Rendering
Scene
Scripting
Shader
Shader Graph
Testing
Terrain
Text Core
TextMeshPro
Timeline
UI Toolkit
Visual Scripting
Vulkan Settings
Physics Material
GUI Skin
Custom Font
D3D12 Settings
Input Actions
Assets
Scenes
Settings
TutorialInfo
InputSyste...
Readme
Assets


Assets
Scenes
Settings
TutorialInfo
InputSyste...
NewMonoBeha
Readme


Code1


File Edit View Git Project Build Debug Test Tools Extensions Window Help Search labY1
Debug Any CPU Assembly-CSharp-Editor Attach to Unity
Use your VS Code Copilot account
Seamlessly extend your Copilot experience to Visual Studio
Sign in
Code1.cs
Assembly-CSharp Code1 Start()
using UnityEngine;
Unity Script | 0 references
public class Code1 : MonoBehaviour
{
// Start is called once before the first execution of Update after the MonoBehaviour is created
Unity Message | 0 references
void Start()
{
}
// Update is called once per frame
Unity Message | 0 references
void Update()
{
}
}
Assets
133 % No issues found
Ln: 1, Ch: 1 SPC CRLF UTF-8
Unity Proj... GitHub C... Solution E... Git Chang...
Ready
Add to Source Control
Select Repository


Unity Project Explorer
Search Unity Project Explorer
Assets
TutorialInfo
Code1.cs

Code1.cs

Assembly-CSharp | Code1 | Start()

```
1   using UnityEngine;
2
    Unity Script | 0 references
3   public class Code1 : MonoBehaviour
4   {
5       // Start is called once before the first execution of Update after the MonoBehaviour is created
        Unity Message | 0 references
6       void Start()
7       {
8
9       }
10
11      // Update is called once per frame
        Unity Message | 0 references
12      void Update()
13      {
14
15      }
16  }
17
```

void Start()
ทำครั้งเดียวเมื่อ GameObject ถูกเรียกใช้

void Update()
ทำงานทุกเฟรม

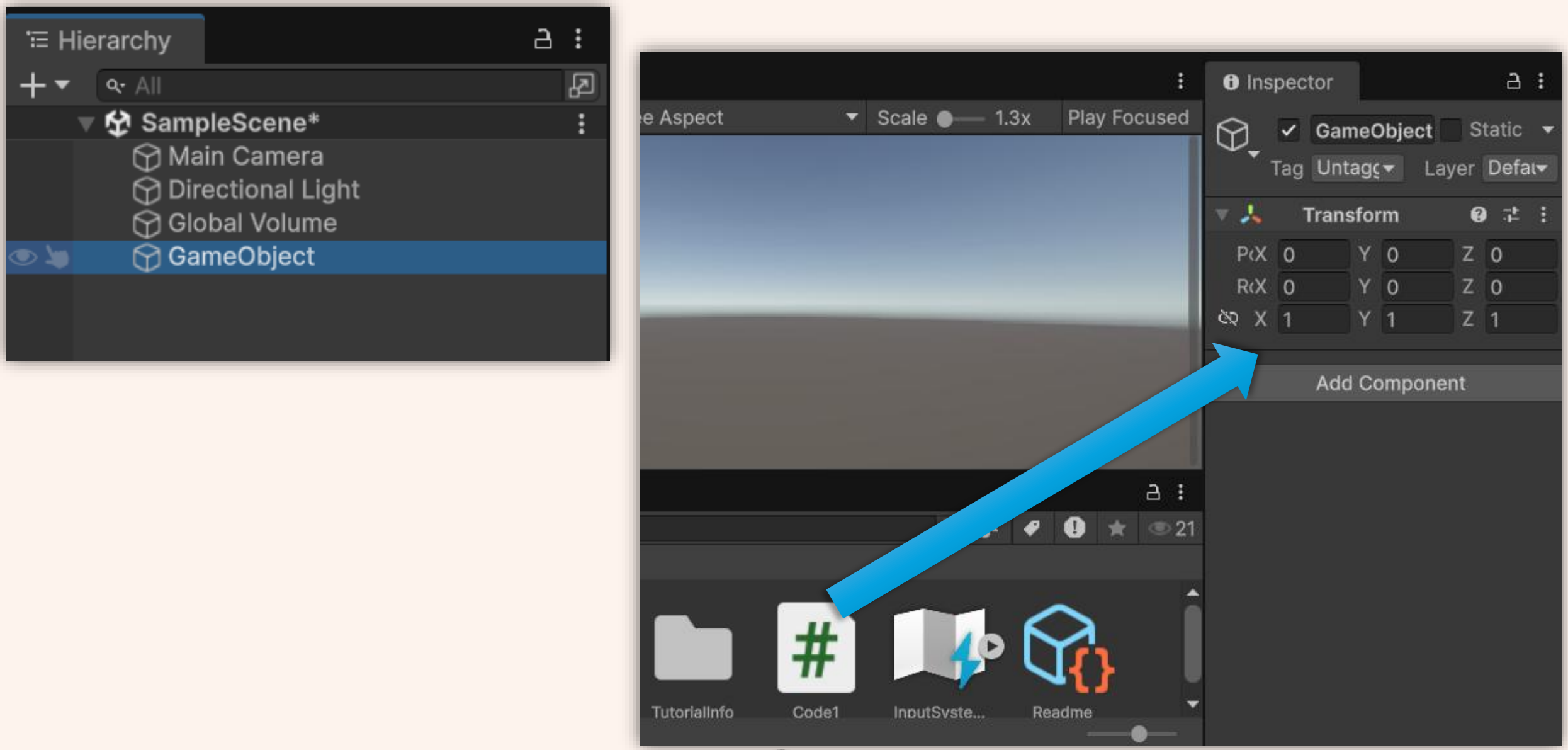
```
1   using UnityEngine;
2
    Unity Script | 0 references
3   public class Code1 : MonoBehaviour
4   {
5       private bool firstTime;
        Unity Message | 0 references
6       void Start()
7       {
8           Debug.Log("Start()....\n");
9           firstTime = true;
10      }
11
12      // Update is called once per frame
        Unity Message | 0 references
13      void Update()
14      {
15          if (firstTime)
16          {
17              firstTime = false;
18              Debug.Log("1st at Update()!!!");
19          }
20      }
21  }
22
```

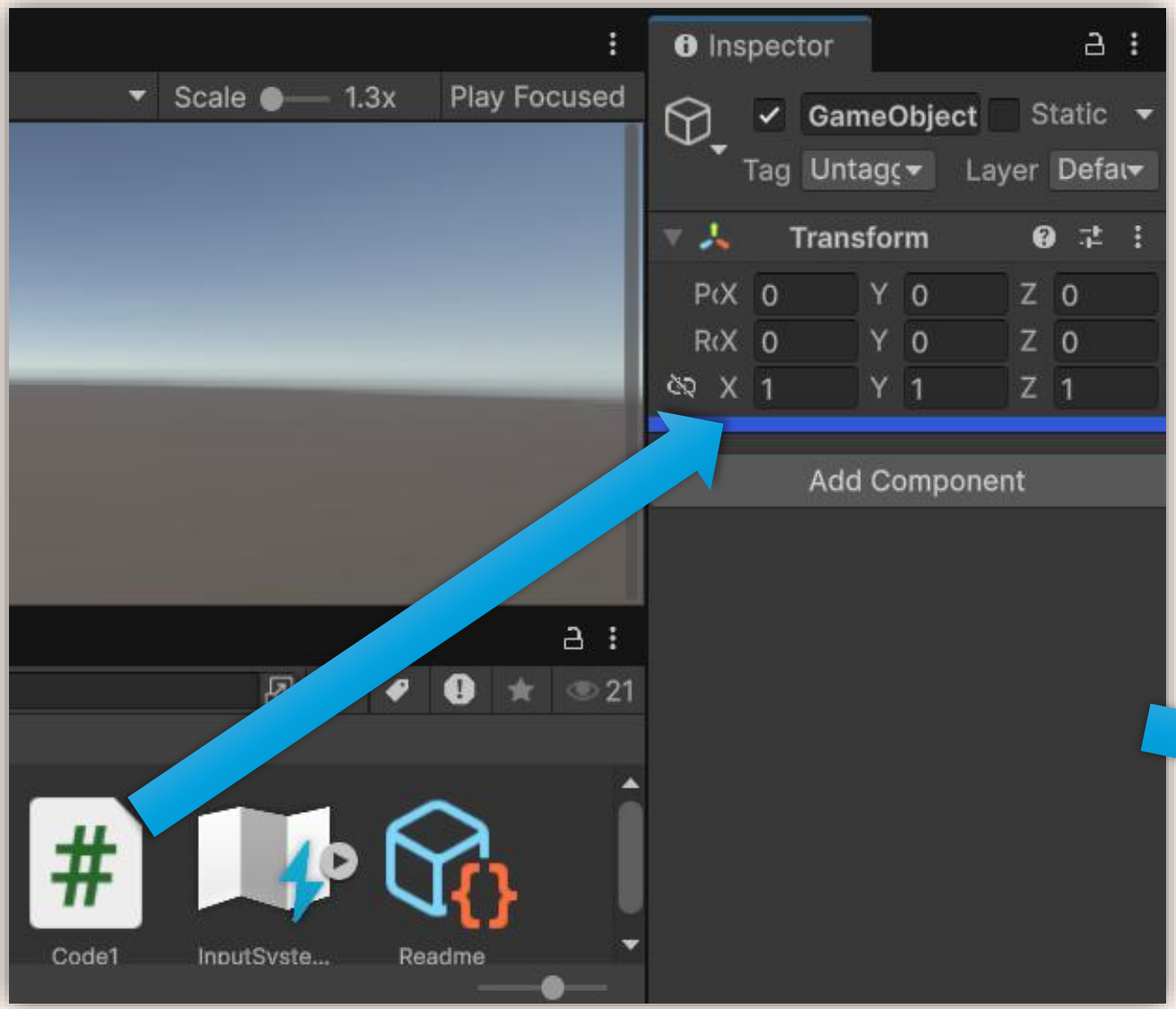
ครั้งแรกที่ GameObejct ถูกเรียกทำงานจะแสดงข้อความ Start()....
หลังจากนั้นกำหนดให้ตัวแปร firstTime เป็น true

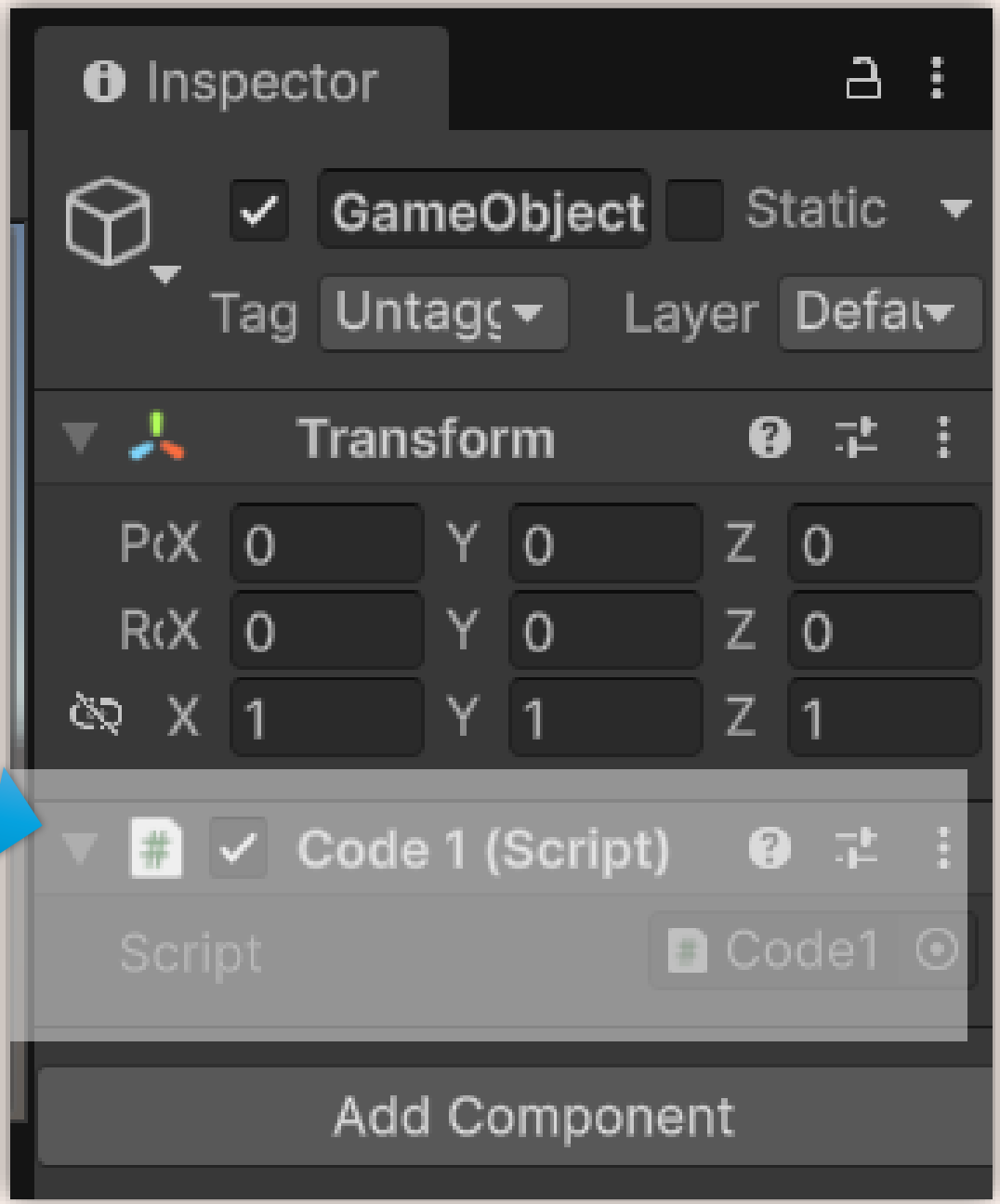
ในแต่ละเฟรมมีการตรวจสอบว่า firstTime เป็นจริงหรือไม่ถ้าใช่จะกำหนดให้ firstTime เป็น false แล้วแสดงข้อความ 1st at Update()!!!

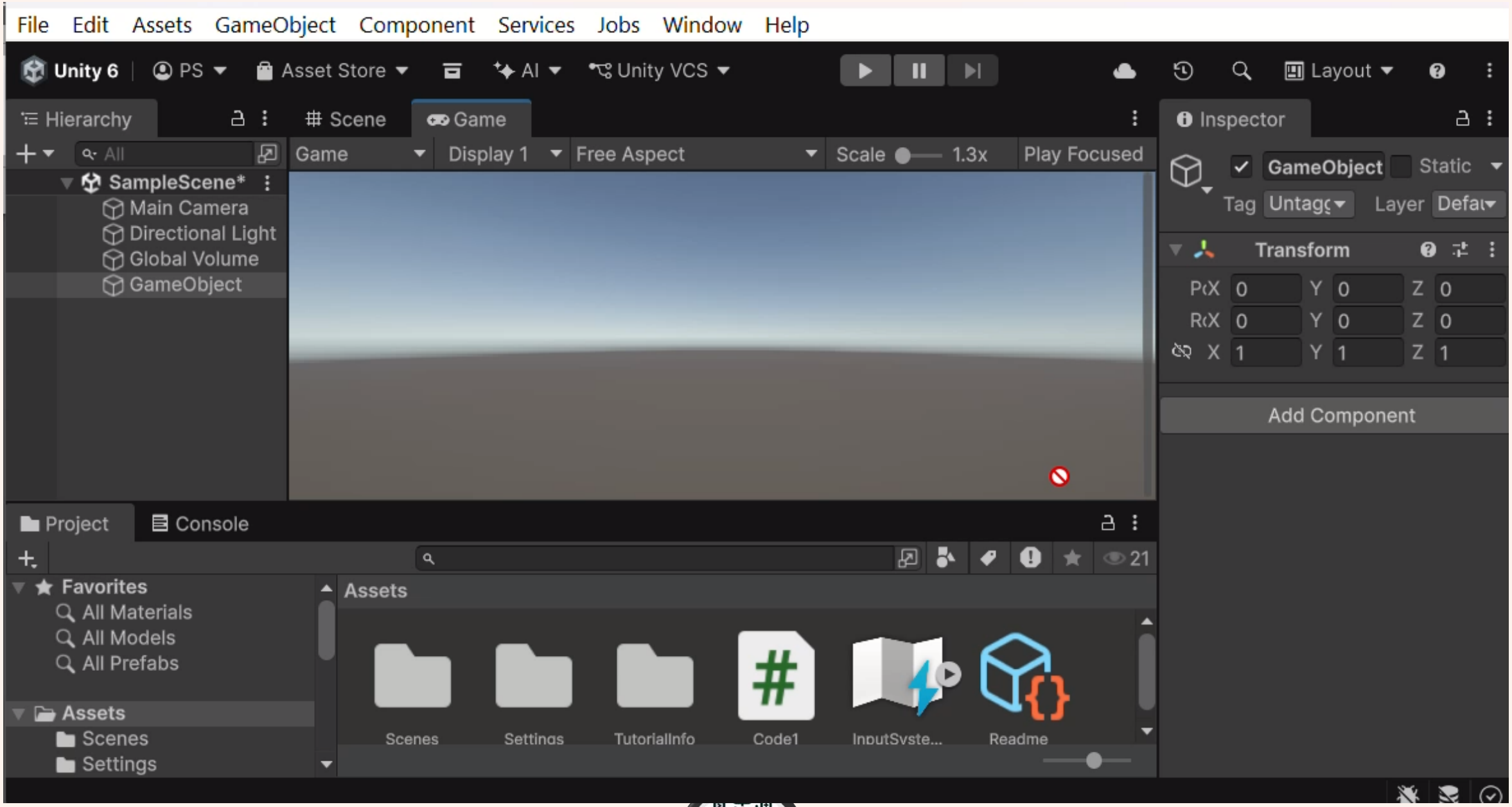
กลับไปที่ Unity แล้วลาก Script ไปวางใน GameObject

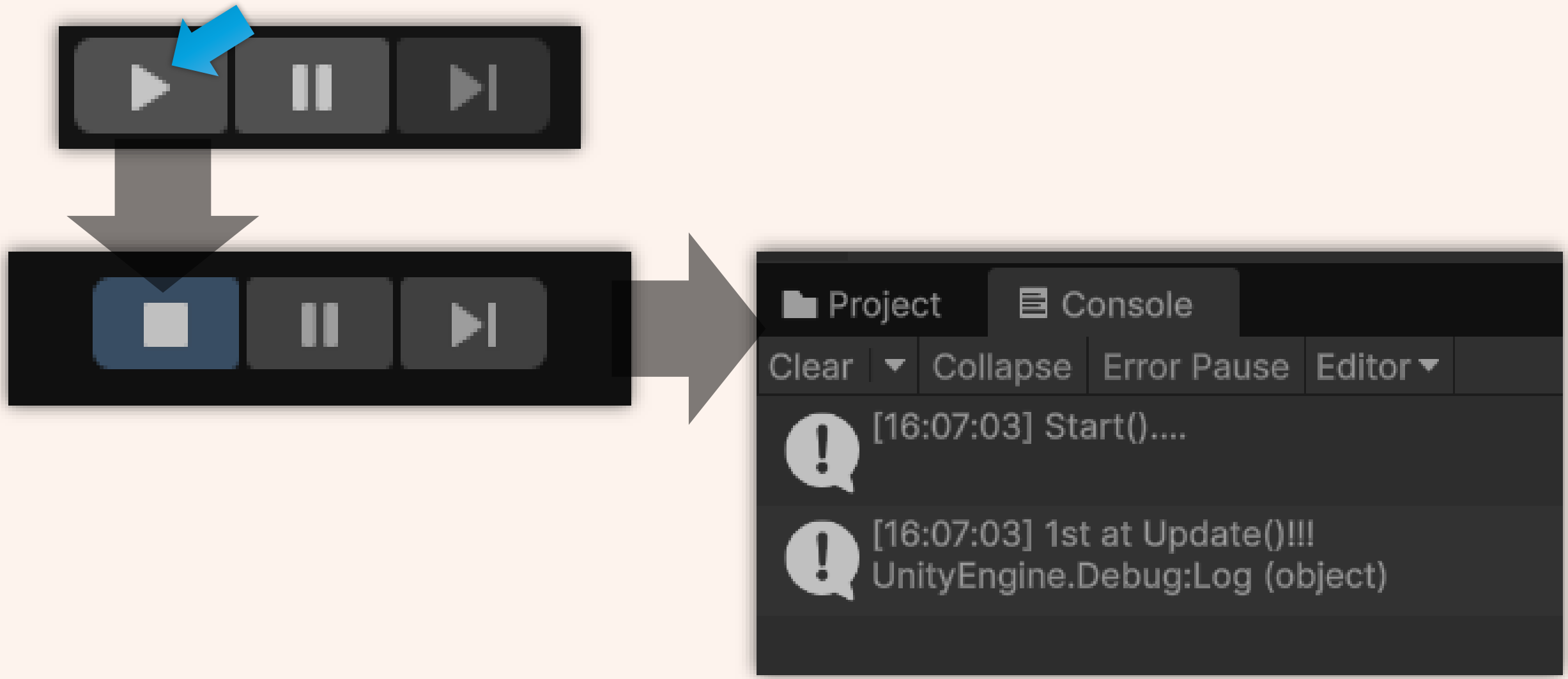
แต่ ต้องดูให้ดีว่าไม่มีข้อผิดพลาด ถ้าผิดพลาด เช่น พิมพ์ผิดจะต้องแก้ไขให้ถูกต้องก่อน


Hierarchy
All
SampleScene*
Main Camera
Directional Light
Global Volume
GameObject
Aspect
Scale
1.3x
Play Focused
Inspector
GameObject
Static
Tag
Layer
Transform
P X 0 Y 0 Z 0
R X 0 Y 0 Z 0
X 1 Y 1 Z 1
Add Component
21
TutorialInfo
Code1
InputSyste...
Readme


Scale 1.3x
Play Focused
Inspector
GameObject
Static
Tag
Layer
Transform
P X 0 Y 0 Z 0
R X 0 Y 0 Z 0
X 1 Y 1 Z 1
Add Component
Code1
InputSyste...
Readme


Inspector
GameObject
Static
Tag
Layer
Transform
P X 0 Y 0 Z 0
R X 0 Y 0 Z 0
X 1 Y 1 Z 1
Code 1 (Script)
Script
Code1
Add Component


File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
PS
Asset Store
AI
Unity VCS
Layout
Hierarchy
Scene
Game
Inspector
All
SampleScene*
Main Camera
Directional Light
Global Volume
GameObject
Game
Display 1
Free Aspect
Scale
1.3x
Play Focused
GameObject
Static
Tag
Layer
Transform
P X 0 Y 0 Z 0
R X 0 Y 0 Z 0
X 1 Y 1 Z 1
Add Component
Project
Console
21
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Settings
Assets
Scenes
Settings
TutorialInfo
Code1
InputSyste...
Readme


Project
Console
Clear
Collapse
Error Pause
Editor
[16:07:03] Start()....
[16:07:03] 1st at Update()!!!
UnityEngine.Debug:Log (object)

ตอนนี้พร้อมสำหรับการเริ่มต้นเขียน C# กันแล้ว

ตัวแปร, ชนิดข้อมูล และ Debug.Log

ตัวแปรนั้นเป็นการจองหน่วยความจำของคอมพิวเตอร์เพื่อจัดเก็บตัวเลขลงในหน่วยความจำที่จองไว้

แต่ถ้าหน่วยความจำไม่เพียงพอจะเกิดปัญหา not enough memory ที่ทำให้โปรแกรมไม่สามารถทำงานต่อไปได้

การประกาศตัวแปร

- Data_type variable_name;
- Data_type variable_name = value;

ชนิดข้อมูล (Data Type)

ชนิดข้อมูลเป็นการกำหนดประเภทของข้อมูลที่เราจะเก็บไว้ในตัวแปร เช่น จำนวนเต็ม, จำนวนทศนิยม, ข้อความ หรือค่าบูลีน (จริงหรือเท็จ) การกำหนดชนิดข้อมูลที่ถูกต้องจะช่วยให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพและหลีกเลี่ยงข้อผิดพลาด

ชนิดข้อมูลพื้นฐาน

- int จำนวนเต็ม (เช่น 1, -10, 0)
- float, double จำนวนทศนิยม (เช่น 3.14, -0.5)
- char อักขระตัวเดียว (เช่น 'A', 'b', '9')
- string ข้อความ (เช่น "Hello", "World")
- bool ค่าความจริง (true หรือ false)

```
int age; // ประกาศตัวแปร age เพื่อเก็บค่าจำนวนเต็ม
string name; // ประกาศตัวแปร name เพื่อเก็บค่าข้อความ
bool isStudent; // ประกาศตัวแปร isStudent เพื่อเก็บค่าความจริง
```

ค่าคงที่เป็นการอ้างอิงชื่อเพื่อใช้แทนค่าตัวเลขทำให้สะดวกในการอ้างอิงหรือเปลี่ยนแปลงค่าแล้วส่งผลให้ทุกชื่อที่อ้างอิงนั้นเปลี่ยนเป็นค่าเดียวกันทั้งหมด

กำหนดค่า

ชื่อตัวแปร = ค่า;

```
age = 20;
name = "GameDev PBRU";
isStudent = true;
```

การประกาศค่าคงที่

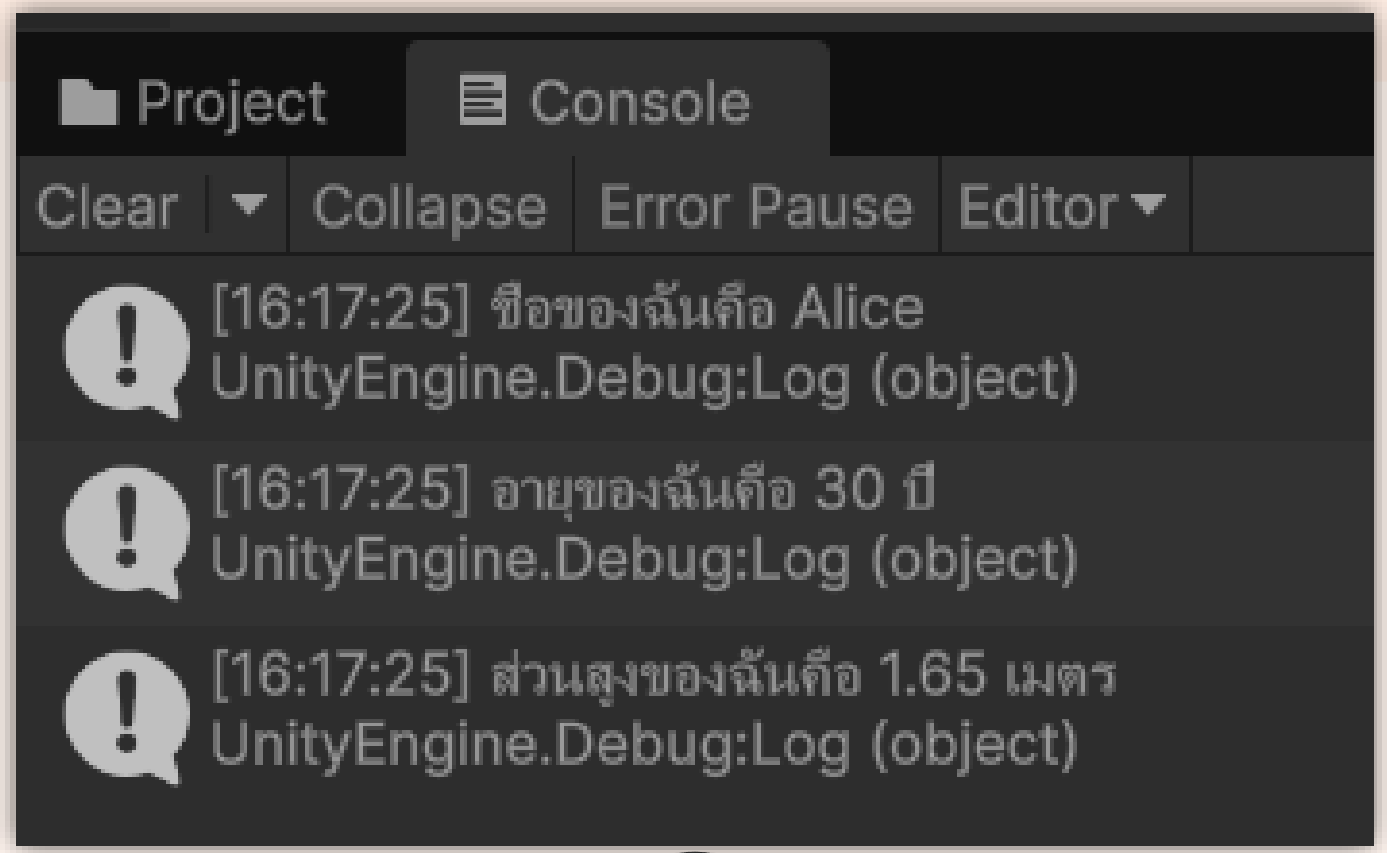
```
const Data_type   constant_name = value;
```

```
const   float   pi = (float)3.1419526;
```

```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Code1 : MonoBehaviour
{
    Unity Message | 0 references
    void Start()
    {
        // ประกาศตัวแปร
        string name = "Alice";
        int age = 30;
        double height = 1.65;

        // แสดงผล
        Debug.Log("ชื่อของฉันคือ " + name);
        Debug.Log("อายุของฉันคือ " + age + " ปี");
        Debug.Log("ส่วนสูงของฉันคือ " + height + " เมตร");
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
}
```

```
void Start()
{
    // ประกาศตัวแปร
    string name = "Alice";
    int age = 30;
    double height = 1.65;

    // แสดงผล
    Debug.Log("ชื่อของฉันคือ " + name);
    Debug.Log("อายุของฉันคือ " + age + " ปี");
    Debug.Log("ส่วนสูงของฉันคือ " + height + " เมตร");
}
```

กลับไปที่ Unity แล้วรันโปรแกรม



นิพจน์และตัวดำเนินการ

โอเปอเรเตอร์ทางคณิตศาสตร์

- การบวก (+): นำค่าสองค่ามาบวกกัน
- การลบ (-): นำค่าสองค่ามาลบกัน
- การคูณ (*): นำค่าสองค่ามาคูณกัน
- การหาร (/): นำค่าสองค่ามาหารกัน
- เศษจากการหาร (%): หาเศษจากการหาร
- เพิ่มค่า (++): เพิ่มค่าตัวแปรขึ้น 1
- ลดค่า (--): ลดค่าตัวแปรลง 1

```
int x = 10;
int y = 5;

int sum = x + y; // sum จะมีค่าเท่ากับ 15
int difference = x - y; // difference จะมีค่าเท่ากับ 5
int product = x * y; // product จะมีค่าเท่ากับ 50
int quotient = x / y; // quotient จะมีค่าเท่ากับ 2
int remainder = x % y; // remainder จะมีค่าเท่ากับ 0

x++; // x จะมีค่าเท่ากับ 11
y--; // y จะมีค่าเท่ากับ 4
```

โอเปอเรเตอร์เปรียบเทียบ

- เท่ากับ (==): ตรวจสอบว่าค่าสองค่าเท่ากันหรือไม่
- ไม่เท่ากับ (!=): ตรวจสอบว่าค่าสองค่าไม่เท่ากัน
- มากกว่า (>): ตรวจสอบว่าค่าทางซ้ายมากกว่าค่าทางขวาหรือไม่
- น้อยกว่า (<): ตรวจสอบว่าค่าทางซ้ายน้อยกว่าค่าทางขวาหรือไม่
- มากกว่าหรือเท่ากับ (>=): ตรวจสอบว่าค่าทางซ้ายมากกว่าหรือเท่ากับค่าทางขวาหรือไม่
- น้อยกว่าหรือเท่ากับ (<=): ตรวจสอบว่าค่าทางซ้ายน้อยกว่าหรือเท่ากับค่าทางขวาหรือไม่

```
int age = 20;
if (age >= 18)
{
    Debug.Log("คุณมีอายุครบ 18 ปีแล้ว");
}
```

โอเปอเรเตอร์ตรรกะ

- และ (&&): ทั้งสองเงื่อนไขต้องเป็นจริง
- หรือ (||): อย่างน้อยหนึ่งเงื่อนไขต้องเป็นจริง
- ไม่ (!): เปลี่ยนค่าความจริงเป็นตรงกันข้าม

```
bool isRaining = true;
bool isSunny = false;

if (isRaining && !isSunny)
{
    Debug.Log("วันนี้ฝนตกและไม่แดดออก");
}
```

โอเปอเรเตอร์แบบบิต (Bitwise Operator)

- And (&): ทำการ AND แบบบิต
- Or (|): ทำการ OR แบบบิต
- XOR (^): ทำการ XOR แบบบิต
- Not (~): ทำการ NOT แบบบิต
- Shift left (<<): เลื่อนบิตไปทางซ้าย
- Shift right (>>): เลื่อนบิตไปทางขวา

```
int x = 5; // ในระบบเลขฐานสองคือ 0101
int y = 3; // ในระบบเลขฐานสองคือ 0011
int result = x & y;
// result จะมีค่าเท่ากับ 1 (ในระบบเลขฐานสองคือ 0001)
```

โอเปอเรเตอร์อื่นๆ

- null-coalescing operator: ?? (ใช้กับ null)
- conditional operator: ?: (ใช้สำหรับเขียน if-else ในบรรทัดเดียว)
- member access operator: . (ใช้เข้าถึงสมาชิกของ class หรือ object)
- index operator: [] (ใช้เข้าถึงสมาชิกของ array)

```
int age = 25;
bool isAdult = age >= 18;
string message = isAdult ? "คุณเป็นผู้ใหญ่แล้ว" : "คุณยังไม่เป็นผู้ใหญ่";
Console.WriteLine(message);
```

คำสั่งเงื่อนไข (if, switch) สำหรับ AI & Game State

คำสั่งควบคุม

คำสั่งควบคุมในภาษา C# ช่วยให้เราสามารถควบคุมการทำงานของโปรแกรมได้อย่างมีประสิทธิภาพ โดยคำสั่งเหล่านี้จะทำหน้าที่ตัดสินใจว่าจะดำเนินการส่วนใดของโค้ดต่อไปขึ้นอยู่กับเงื่อนไขที่กำหนด

if-else
ใช้สำหรับตัดสินใจ

for
ใช้สำหรับวนลูปจำนวนครั้งที่แน่นอน

while
ใช้สำหรับวนลูปจนกว่าเงื่อนไขจะเป็นเท็จ

do-while
ใช้สำหรับวนลูปอย่างน้อย 1 ครั้ง

คำสั่ง if-else

ใช้สำหรับตัดสินใจว่าจะดำเนินการส่วนใดส่วนหนึ่งของโค้ด โดยขึ้นอยู่กับเงื่อนไขที่กำหนด

```
if (เงื่อนไข)
{
    // ถ้าเงื่อนไขเป็นจริง จะทำงานในนี้
}
else
{
    // ถ้าเงื่อนไขเป็นเท็จ จะทำงานในนี้
}
```

```
int number = 10;

if (number % 2 == 0)
{
    Debug.Log("เลข {0} เป็นเลขคู่", number);
}
else
{
    Debug.Log("เลข {0} เป็นเลขคี่", number);
}
```

การวนซ้ำ (for, while, foreach)

คำสั่ง for

ใช้สำหรับวนลูปทำซ้ำชุดคำสั่งจำนวนครั้งที่แน่นอน

```
for (เริ่มต้น; เงื่อนไข; เพิ่มค่า)
{
    // คำสั่งที่ต้องการทำซ้ำ
}
```

```
for (int i = 1; i <= 10; i++)
{
    Debug.Log(i);
}
```

คำสั่ง while

ใช้สำหรับวนลูปทำซ้ำชุดคำสั่งจนกว่าเงื่อนไขจะเป็นเท็จ

```
while (เงื่อนไข)
{
    // คำสั่งที่ต้องการทำซ้ำ
}
```

```
int sum = 0;
int i = 1;

while (i <= 5)
{
    sum += i;
    i++;
}

Debug.Log("ผลรวม = " + sum);
```

คำสั่ง foreach

ใช้สำหรับวนลูปอ่านข้อมูลจาก Collection ต่าง ๆ เช่น Array, List, Dictionary, HashSet หรือแม้แต่ Children ของ GameObject โดยไม่ต้องจัดการตัวแปร Index เอง

```
foreach (ชนิดข้อมูล ตัวแปร in Collection)
{
    // ทำงานกับข้อมูลแต่ละตัว
}
```

```
string[] names = { "Cat", "Dog", "Bird" };

foreach (string name in names)
{
    Debug.Log(name);
}
```

```
using UnityEngine;

public class ArrayExample : MonoBehaviour
{
    void Start()
    {
        int[] scores = { 100, 200, 300, 400 };

        foreach (int score in scores)
        {
            Debug.Log($"Score = {score}");
        }
    }
}
```

คำแนะนำสำหรับ Unity

ใช้ foreach สำหรับ

- อ่านข้อมูล
- วนลูป Child Objects
- วนลูป Components
- วนลูป ScriptableObject Lists
- ระบบ Inventory
- ระบบ Quest
- ระบบ UI

คำแนะนำสำหรับ Unity

ใช้ for สำหรับ

- ต้องการ Index
- เพิ่ม/ลบสมาชิกใน Collection
- ลูปจำนวนมากใน Update()
- ระบบ AI จำนวนมาก
- ระบบ Physics หรือ Simulation ที่ต้อง Optimize

อาร์เรย์ (Arrays) และการจัดการชุดข้อมูลคงที่

Array

โครงสร้างข้อมูลที่ใช้เก็บข้อมูลหลายค่าในตัวแปรเดียวกัน โดยข้อมูลทุกตัวใน Array ต้องเป็นชนิดเดียวกัน และมีขนาดคงที่ (Fixed Size) ตั้งแต่ตอนสร้าง

คะแนนนักเรียน 5 คน

Index:	0	1	2	3	4
Value:	80	75	92	68	85

การประกาศ Array

ประเภทข้อมูล[] ชื่อตัวแปร = new ประเภทข้อมูล[จำนวน];

เช่น

```
int[] scores = new int[5];
```

```
int[] scores = { 80, 75, 92, 68, 85 };
```

หรือ

```
int[] scores = new int[]
{
    80,
    75,
    92,
    68,
    85
};
```

การเข้าถึงข้อมูล

```
int[] scores = { 80, 75, 92 };


Debug.Log(scores[0]);
Debug.Log(scores[1]);
Debug.Log(scores[2]);
```

```
string[] animals =
{
    "Cat",
    "Dog",
    "Rabbit",
    "Tiger"
};
Debug.Log("สมาชิกใน Array");
Debug.Log(animals[0]);
Debug.Log(animals[1]);
Debug.Log(animals[2]);
Debug.Log(animals[3]);
```

การเปลี่ยนค่าใน Array

```
int[] scores = { 50, 60, 70 };
scores[1] = 100;
Debug.Log(scores[0]);
Debug.Log(scores[1]);
Debug.Log(scores[2]);
```

```
string[] weapons =
{
    "Sword",
    "Bow",
    "Gun",
    "Axe"
};

for (int i = 0; i < weapons.Length; i++)
{
    Debug.Log(weapons[i]);
}
```

```
string[] weapons =
{
    "Sword",
    "Bow",
    "Gun",
    "Axe"
};

foreach (string weapon in weapons)
{
    Debug.Log(weapon);
}
```

คุณสมบัติ Length

ใช้ดูจำนวนสมาชิกใน Array

```
int[] numbers =
{
    10,
    20,
    30,
    40,
    50
};

Debug.Log(numbers.Length);
```

คำนวณผลรวม

```
int[] scores =
{
    80,
    75,
    92,
    68,
    85
};
int sum = 0;
foreach (int score in scores)
{
    sum += score;
}
Debug.Log("Total = " + sum);
```

หาค่าเฉลี่ย

```
int[] scores =
{
    80,
    75,
    92,
    68,
    85
};
int total = 0;
foreach (int score in scores)
{
    total += score;
}
float average = (float)total / scores.Length;
Debug.Log("Average = " + average);
```

Array สองมิติ

```
int[,] map =
{
    {1, 1, 1},
    {1, 0, 1},
    {1, 1, 1}
};
Debug.Log(map[1,1]);
```

Inventory อย่างง่าย

```
string[] inventory =
{
    "Potion",
    "Sword",
    "Shield",
    "Key"
};
Debug.Log("Inventory");
for (int i = 0; i < inventory.Length; i++)
{
    Debug.Log(
        $"Slot {i}: {inventory[i]}"
    );
}
```

Array เหมาะสำหรับ

- เก็บข้อมูลจำนวนคงที่
- เก็บคะแนน
- เก็บตำแหน่ง Spawn Point
- เก็บ Enemy ที่กำหนดไว้ล่วงหน้า
- เก็บรายการไอเท็มแบบคงที่
- ระบบ Tile Map ขนาดคงที่

ตัวอย่าง tic-tac-toe

Tic-Tac-Toe

เป็นการผลัดกันเล่นระหว่างผู้เล่น (Human Player) และคอมพิวเตอร์ (Computer Player) ทำการผลัดกันเลือกตำแหน่งบนกระดานขนาด 3×3 จนกว่าจะเกิดผู้ชนะ หรือไม่มีตำแหน่งว่างเหลือบนกระดาน ถือเป็นเกมที่ใช้โครงสร้างข้อมูลแบบ Array 2 มิติในการเก็บสถานะของเกมและควบคุมลำดับการเล่นตามกติกาอย่างเป็นระบบ

Tic-Tac-Toe

1. InitializeBoard() ทำหน้าที่กำหนดให้ทุกช่องของกระดานมีค่าเป็น ' '

void InitializeBoard(char[,] board)

2. DisplayBoard() ทำหน้าที่แสดงสถานะปัจจุบันของกระดานบน Console ในรูปแบบ 3×3

void DisplayBoard(char[,] board)

3. PlayerMove() ทำหน้าที่รับข้อมูลตำแหน่ง (row, column) จากผู้เล่น โดยต้องตรวจสอบว่า ตำแหน่งอยู่ในช่วงที่ถูกต้อง (0–2) และช่องดังกล่าวว่างอยู่

void PlayerMove(char[,] board)

4. ComputerMove() ทำหน้าที่ให้คอมพิวเตอร์เลือกตำแหน่งว่างแบบสุ่ม หรือเช็กช่องว่างทีละตำแหน่ง

void ComputerMove(char[,] board)

5. CheckWinner() ทำหน้าที่ตรวจสอบว่ามีผู้ชนะหรือไม่ โดยตรวจสอบแถวทั้งสาม คอลัมน์ทั้งสาม และเส้นทแยงมุมสองเส้น ซึ่งคืนค่ากลับเป็น 'X' ถ้าผู้เล่นชนะ, 'O' ถ้าคอมพิวเตอร์ชนะ หรือ ' ' ถ้ายังไม่มีผู้ชนะ

char CheckWinner(char[,] board)

6. IsBoardFull() ทำหน้าที่ตรวจสอบว่ากระดานเต็มหรือไม่

bool IsBoardFull(char[,] board)

Tic-Tac-Toe

1. เริ่มต้นโปรแกรม
2. เรียก InitializeBoard()
3. วนลูปเกมดังนี้
    1. แสดงกระดานด้วยการเรียกใช้เมธอด DisplayBoard()
    2. ผู้เล่นทำการเดินหมากด้วยการเรียกใช้เมธอด PlayerMove()
    3. ตรวจสอบผลลัพธ์ด้วยการเรียกใช้เมธอด CheckWinner()
    4. หากมีผู้ชนะให้จบเกม
    5. หากกระดานเต็มแสดงว่าจบเกมแบบเสมอกัน
    6. คอมพิวเตอร์ทำการเดินหมากด้วยการเรียกใช้เมธอด ComputerMove()
    7. ตรวจสอบผลลัพธ์ด้วยการเรียกใช้เมธอด CheckWinner()
    8. หากมีผู้ชนะ ให้จบเกม
4. แสดงผลลัพธ์ (ผู้ชนะหรือเสมอ)
5. จบโปรแกรม

Tic-Tac-Toe

ปุ่ม 1 แทนตำแหน่ง (0,0)	ปุ่ม 2 แทนตำแหน่ง (0,1)	ปุ่ม 3 แทนตำแหน่ง (0,2)
ปุ่ม 4 แทนตำแหน่ง (1,0)	ปุ่ม 5 แทนตำแหน่ง (1,1)	ปุ่ม 6 แทนตำแหน่ง (1,2)
ปุ่ม 7 แทนตำแหน่ง (2,0)	ปุ่ม 8 แทนตำแหน่ง (2,1)	ปุ่ม 9 แทนตำแหน่ง (2,2)

```
using UnityEngine;
0 references
public class TicTacToe : MonoBehaviour
{
    31 references
    private char[,] board = new char[3, 3];
    5 references
    private bool playerTurn = true;
    5 references
    private bool gameOver = false;
    0 references
    void Start()
    {
        InitializeBoard();
        DisplayBoard();
        Debug.Log("Game Start - Player (X) begins.");
    }
```

```
13      void Update()
14      {
15          if (gameOver)
16          {
17              return;
18          }
19          if (playerTurn)
20          {
21              if (Input.GetKeyDown(KeyCode.Alpha1)) PlayerMove(0, 0);
22              if (Input.GetKeyDown(KeyCode.Alpha2)) PlayerMove(0, 1);
23              if (Input.GetKeyDown(KeyCode.Alpha3)) PlayerMove(0, 2);
24              if (Input.GetKeyDown(KeyCode.Alpha4)) PlayerMove(1, 0);
25              if (Input.GetKeyDown(KeyCode.Alpha5)) PlayerMove(1, 1);
26              if (Input.GetKeyDown(KeyCode.Alpha6)) PlayerMove(1, 2);
27              if (Input.GetKeyDown(KeyCode.Alpha7)) PlayerMove(2, 0);
28              if (Input.GetKeyDown(KeyCode.Alpha8)) PlayerMove(2, 1);
29              if (Input.GetKeyDown(KeyCode.Alpha9)) PlayerMove(2, 2);
30          }
31      }
```

```
32      void InitializeBoard()
33      {
34          for (int i = 0; i < 3; i++)
35          {
36
37              for (int j = 0; j < 3; j++)
38              {
39                  board[i, j] = ' ';
40              }
41          }
42      }
```

```
43      void DisplayBoard()
44      {
45          Debug.Log("-------------");
46          for (int i = 0; i < 3; i++)
47          {
48              string row = "| ";
49              for (int j = 0; j < 3; j++)
50              {
51                  row += board[i, j] + " | ";
52              }
53              Debug.Log(row);
54              Debug.Log("-------------");
55          }
56      }
```

```
57      void PlayerMove(int row, int col)
58      {
59          if (!playerTurn || gameOver)
60          {
61              return;
62          }
63          if (board[row, col] == ' ')
64          {
65              board[row, col] = 'X';
66              Debug.Log($"Player places X at ({row},{col})");
67              playerTurn = false;
68              CheckState();
69          }
70          else
71          {
72              Debug.Log("Invalid move. Position already taken.");
73          }
74      }
```

```
75      void ComputerMove()
76      {
77          Debug.Log("Computer is thinking...");
78          int row, col;
79          System.Random rand = new System.Random();
80          while (true)
81          {
82              row = rand.Next(0, 3);
83              col = rand.Next(0, 3);
84              if (board[row, col] == ' ')
85              {
86                  board[row, col] = 'O';
87                  Debug.Log($"Computer places O at ({row},{col})");
88                  break;
89              }
90          }
91          playerTurn = true;
92          CheckState();
93      }
```

```
94      void CheckState()
95      {
96          char winner = CheckWinner();
97          DisplayBoard();
98          if (winner == 'X')
99          {
100             Debug.Log("Player Wins!");
101             gameOver = true;
102             return;
103         }
104         if (winner == 'O')
105         {
106             Debug.Log("Computer Wins!");
107             gameOver = true;
108             return;
109         }
```

```
110         if (IsBoardFull())
111         {
112             Debug.Log("Draw!");
113             gameOver = true;
114             return;
115         }
116         if (!playerTurn)
117         {
118             // ดีเลย์คอม (optional)
119             Invoke("ComputerMove", 0.5f);
120         }
121     }
```

```
122     char CheckWinner()
123     {   // Rows and Columns
124         for (int i = 0; i < 3; i++)
125         {
126             if (board[i, 0] != ' ' &&
127                 board[i, 0] == board[i, 1] &&
128                 board[i, 1] == board[i, 2])
129                 return board[i, 0];
130 
131             if (board[0, i] != ' ' &&
132                 board[0, i] == board[1, i] &&
133                 board[1, i] == board[2, i])
134                 return board[0, i];
135         }
```

```
136         // Diagonals
137         if (board[0, 0] != ' ' &&
138             board[0, 0] == board[1, 1] &&
139             board[1, 1] == board[2, 2])
140             return board[0, 0];
141         if (board[0, 2] != ' ' &&
142             board[0, 2] == board[1, 1] &&
143             board[1, 1] == board[2, 0])
144             return board[0, 2];
145         return ' ';
146     }
```

```
147     bool IsBoardFull()
148     {
149         foreach (char c in board)
150             if (c == ' ') return false;
151         return true;
152     }
153 }
```

Mine-Sweep

ตัวอย่างโปรแกรมเพื่อสุ่มระเบิดในพื้นที่ขนาด 8x8 โดยให้มีระเบิดจำนวน 10 ลูก พร้อมทั้งคำนวณจำนวนระเบิดที่อยู่รอบตัวของแต่ละจุด โปรแกรมจะใช้เมทริกซ์ 2 มิติ (int[,]) ในการเก็บข้อมูล โดยที่ ค่า -1 หมายถึง ตำแหน่งนั้นคือระเบิด และค่า 0 ถึง 8 หมายถึง จำนวนระเบิดที่อยู่รอบ ๆ จุดนั้น

```
1  using UnityEngine;
2  using System.Collections.Generic;
   0 references
3  public class MinesweeperGenerator : MonoBehaviour
4  {
5      // กำหนดขนาดกระดานและจำนวนระเบิด
       7 references
6      private const int WIDTH = 8;
       6 references
7      private const int HEIGHT = 8;
       2 references
8      private const int NUMBER_OF_BOMBS = 10;
9      // แผนที่กระดาน (Map Grid) โดย -1 คือระเบิด, 0-8 คือจำนวนระเบิดรอบข้าง
       6 references
10     private int[,] gameGrid = new int[WIDTH, HEIGHT];
```

```
11      void Start()
12      {
13          Debug.Log("--- เริ่มสร้างกระดาน Minesweeper ขนาด 8x8 (10 ระเบิด)");
14          // 1. กำหนดค่าเริ่มต้นให้กับกระดานเป็น 0 ทั้งหมด
15          InitializeGrid();
16          // 2. วางระเบิดแบบสุ่ม
17          PlaceBombs();
18          // 3. คำนวณจำนวนระเบิดที่อยู่รอบ ๆ
19          CalculateAdjacentBombs();
20          // 4. แสดงผลลัพธ์ใน Console
21          PrintGrid();
22          Debug.Log("--- สร้างกระดานเสร็จสมบูรณ์ ---");
23      }
```

```
24  // กำหนดค่าเริ่มต้นของกระดานให้เป็น 0 ทั้งหมด
        1 reference
25      private void InitializeGrid()
26      {
27          // การสร้าง int[,] จะเริ่มต้นด้วย 0 อยู่แล้ว แต่ทำซ้ำเพื่อความชัดเจน
28          for (int x = 0; x < WIDTH; x++)
29          {
30              for (int y = 0; y < HEIGHT; y++)
31              {
32                  gameGrid[x, y] = 0;
33              }
34          }
35      }
```

```
36  // วางระเบิดจำนวน 10 ลูกแบบสุ่มบนกระดาน (-1)
        1 reference
37      private void PlaceBombs()
38      {
39          int bombsPlaced = 0;
40          // ใช้ List เพื่อเก็บตำแหน่งที่ว่างทั้งหมด
41          List<Vector2Int> availablePositions = new List<Vector2Int>();
42          for (int x = 0; x < WIDTH; x++)
43          {
44              for (int y = 0; y < HEIGHT; y++)
45              {
46                  availablePositions.Add(new Vector2Int(x, y));
47              }
48          }
```

```
49          while (bombsPlaced < NUMBER_OF_BOMBS)
50          {
51              // สุ่มเลือกตำแหน่งจาก List ที่มี
52              int randomIndex = Random.Range(0, availablePositions.Count);
53              Vector2Int bombPos = availablePositions[randomIndex];
54              // วางระเบิด (-1) และลบตำแหน่งนั้นออกจาก List ที่ว่าง
55              gameGrid[bombPos.x, bombPos.y] = -1;
56              availablePositions.RemoveAt(randomIndex);
57              bombsPlaced++;
58          }
59          Debug.Log($"2. วางระเบิด {NUMBER_OF_BOMBS} ลูกเสร็จสมบูรณ์");
60      }
```

```
61  // คำนวณจำนวนระเบิดที่อยู่รอบ ๆ แต่ละจุด
62  // โดยจะใช้ระเบิดแต่ละลูกในการอัปเดตค่าของเพื่อนบ้าน
        1 reference
63      private void CalculateAdjacentBombs()
64      {
65          for (int x = 0; x < WIDTH; x++)
66          {
67              for (int y = 0; y < HEIGHT; y++)
68              {
69                  // ตรวจสอบว่าจุดนี้เป็นระเบิดหรือไม่
70                  if (gameGrid[x, y] == -1)
71                  {
72                      // ถ้าเป็นระเบิด ให้อัปเดตเพื่อนบ้านทั้ง 8 ทิศ
73                      UpdateNeighbors(x, y);
74                  }
75              }
76          }
77          Debug.Log("3. คำนวณจำนวนระเบิดรอบข้างเสร็จสมบูรณ์");
78      }
```

```
79  // เมธอดช่วยในการอัปเดตค่าของเพื่อนบ้านรอบ ๆ จุดที่เป็นระเบิด
        1 reference
80      private void UpdateNeighbors(int bombX, int bombY)
81      {
82          // วนลูปตรวจสอบเพื่อนบ้านในรัศมี 3x3 (รวมตัวเอง)
83          for (int dx = -1; dx <= 1; dx++)
84          {
85              for (int dy = -1; dy <= 1; dy++)
86              {
87                  // ข้ามจุดของระเบิดเอง (dx=0 และ dy=0)
88                  if (dx == 0 && dy == 0)
89                      continue;
90                  int neighborX = bombX + dx;
91                  int neighborY = bombY + dy;
```

```
92                  // ตรวจสอบว่าเพื่อนบ้านอยู่ในขอบเขตของกระดานหรือไม่
93                  if (neighborX >= 0 && neighborX < WIDTH &&
94                      neighborY >= 0 && neighborY < HEIGHT)
95                  {
96                      // ตรวจสอบว่าจุดเพื่อนบ้านนั้นไม่ใช่ระเบิด
97                      if (gameGrid[neighborX, neighborY] != -1)
98                      {
99                          // เพิ่มจำนวนระเบิดรอบข้าง
100                         gameGrid[neighborX, neighborY]++;
101                     }
102                 }
103             }
104         }
105     }
```

```
106  // แสดงผลลัพธ์ของกระดานใน Unity Console
         1 reference
107      private void PrintGrid()
108      {
109          string gridString = "  ";
110          // แสดงหมายเลขคอลัมน์ด้านบน
111          for (int x = 0; x < WIDTH; x++)
112          {
113              gridString += $"| {x} ";
114          }
115          gridString += "|\n";
116          // แสดงเส้นแบ่ง
117          gridString += "-----------------------------------\n";
```

```
118         for (int y = 0; y < HEIGHT; y++)
119         {
120             gridString += $"{y} "; // แสดงหมายเลขแถว
121             for (int x = 0; x < WIDTH; x++)
122             {
123                 int value = gameGrid[x, y];
124                 string cell;
125                 if (value == -1)
126                 {
127                     // ใช้ X หรือ B เพื่อแทนระเบิด (Minesweeper Bomb: X)
128                     cell = " X ";
129                 }
130                 else
131                 {
132                     cell = $" {value} ";
133                 }
134                 gridString += $"|{cell}";
135             }
136             gridString += "|\n";
137         }
138         Debug.Log("4. แผนที่กระดาน Minesweeper:\n" + gridString);
139     }
140 }
```

Q&A