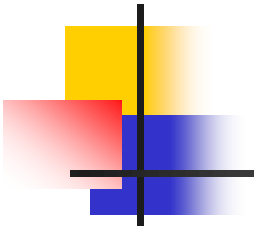
ฟังก์ชันของโปรแกรม

ฟังก์ชันในโปรแกรม (โปรแกรมภาษา C#) มีฟังก์ชันให้ใช้งานอยู่หลากหลายฟังก์ชัน โดยมีรูปแบบเฉพาะ และการเข้าถึงที่มีลักษณะแตกต่างกัน ในบทนี้จะแสดงเนื้อหาในการใช้งานของฟังก์ชันต่างๆ



ประกอบด้วย

- รูปแบบ ขอบเขต การเข้าถึงฟังก์ชัน
- ฟังก์ชันทางคณิตศาสตร์
- ฟังก์ชันเกี่ยวกับเวลา
- ฟังก์ชันอื่นๆ

รูปแบบ ขอบเขต การเข้าถึงฟังก์ชัน

ฟังก์ชัน มีลักษณะเหมือน “โปรแกรมย่อย” ที่ช่วยให้การเขียนโปรแกรมมีความเป็นระเบียบมากขึ้น แบ่งออกเป็นสัดส่วน ลดจำนวนการเขียนโปรแกรมที่ซ้ำกันออกไป ซึ่งเป็นพื้นฐานก่อนการเขียนโปรแกรมเชิงวัตถุ การสร้างฟังก์ชันเพื่อให้มีหน้าที่ทำงานใดงานหนึ่งเท่านั้นให้กับโปรแกรมหลัก “หน้าที่” ต้องมีผู้กระทำหน้าที่(function)กับ ผู้สั่งงานโปรแกรมที่เรียกใช้ฟังก์ชัน

ดังนั้นการสร้าง ฟังก์ชันแบ่งเป็น 2 ส่วน คือ การสร้างฟังก์ชัน และ การเรียกใช้งาน

รูปแบบฟังก์ชัน

```
static <ชนิดตัวแปรส่งออก> <ชื่อfunction> (<ชนิดตัวแปร> <ชื่อ param1>, …)
                {
                        // ส่วนของโปรแกรม
                }
```

ในรูปแบบประกอบด้วย การตั้งชื่อฟังก์ชัน การกำหนดประเภทตัวแปรในการส่งออกจากฟังก์ชัน การรับข้อมูลผ่าน Parameter

รูปแบบฟังก์ชัน

การเรียกใช้ฟังก์ชัน ทำได้โดยการพิมพ์ชื่อฟังก์ชัน แล้วตามด้วย **Parameter** หากมีการส่งค่าออกมา ควรมีการสร้างตัวแปรมารับ โดยมีหลายรูปแบบ เช่น

– ฟังก์ชันที่รับค่าและส่งค่าออก ตัวอย่างเช่น ฟังก์ชันคำนวณหาพื้นที่สี่เหลี่ยม มีการรับค่าตัวแปรความกว้าง(**w**) และความสูง (**h**) และส่งค่าพื้นที่สี่เหลี่ยมกลับ (**return**)

```
static int getRectArea(int w, int h)
{
        return w * h;
}
```

รูปแบบฟังก์ชัน

– ฟังก์ชันที่มีการส่งค่าออกอย่างเดียว ตัวอย่างเช่น ฟังก์ชันการส่งค่าตัวแปร **pi**

```
static double getPi()
{
    return 22/7.0;
}
```

รูปแบบฟังก์ชัน

– ฟังก์ชันที่รับค่าอย่างเดียว ตัวอย่างเช่น ฟังก์ชันคำนวณพื้นที่สี่เหลี่ยม มีการรับค่าตัวแปรความกว้าง(**w**) และความสูง (**h**) แต่ไม่มีการส่งค่ากลับ

```
static void showRectArea(int w, int h)
{
        Console.WriteLine("Area is {0}", (w*h));
}
```

รูปแบบฟังก์ชัน

- ฟังก์ชันที่ไม่มีการรับค่า และส่งค่าออก ตัวอย่างเช่น ฟังก์ชันแสดงเส้น

```
static void showLine()
{
        Console.WriteLine("----------------");
}
```

รูปแบบฟังก์ชัน

ตัวอย่าง โปรแกรมเกมคอมพิวเตอร์ในการสร้างฟังก์ชัน **MultiplyByTwo**

```
using UnityEngine;
using System.Collections;
public class VariablesAndFunctions: MonoBehaviour
{
int myInt= 5;

void Start ()
    {
        myInt= MultiplyByTwo(myInt);
        Debug.Log (myInt);
    }
```

รูปแบบฟังก์ชัน

ตัวอย่าง โปรแกรมเกมคอมพิวเตอร์ในการสร้างฟังก์ชัน **MultiplyByTwo**

```
int MultiplyByTwo(int number)
    {
        int ret;
        ret = number * 2;
        return ret;
    }
}
```

ฟังก์ชันทางคณิตศาสตร์

ฟังก์ชันทางคณิตศาสตร์ ให้ใช้งานโดยอยู่ในคลาสที่ชื่อว่า **mathf**

Pi	**ฟังก์ชันทางคณิตศาสตร์แทนค่า pi เท่ากับ 3.14159265358979**
Abs	**ฟังก์ชันทางคณิตศาสตร์ในการหาค่า absolute**
Exp	**ฟังก์ชันทางคณิตศาสตร์ในการคำนวณค่า e ยกกำลัง (ค่าคงที่ e เท่ากับ 2.71828182845904 ซึ่งเป็นฐานของลอการิทึมธรรมชาติ)**
Max	**ฟังก์ชันทางคณิตศาสตร์ คำนวณค่าที่สูงกว่าระหว่างค่า 2 ค่า หรือมากกว่า**
Min	**ฟังก์ชันทางคณิตศาสตร์ คำนวณค่าที่ต่ำกว่าระหว่างค่า 2 ค่า หรือมากกว่า**

ฟังก์ชันทางคณิตศาสตร์

ฟังก์ชันทางคณิตศาสตร์ ให้ใช้งานโดยอยู่ในคลาสที่ชื่อว่า **mathf**

Pow	**ฟังก์ชันทางคณิตศาสตร์ คำนวณค่ายกกำลัง**
Round	**ฟังก์ชันทางคณิตศาสตร์ แสดงค่าประมาณใกล้จำนวนเต็ม**
Sqrt	**ฟังก์ชันทางคณิตศาสตร์ คำนวณค่ารากที่สอง**
Sin	**ฟังก์ชันทางคณิตศาสตร์ แสดงค่า sine ของมุมในหน่วยองศาเรเดียน**
Cos	**ฟังก์ชันทางคณิตศาสตร์ แปลงเป็นค่า cosine**
Tan	**ฟังก์ชันทางคณิตศาสตร์ แสดงค่า tangent ของมุมในหน่วยองศาเรเดียน**

ฟังก์ชันอื่นๆ

โปรแกรมเกมคอมพิวเตอร์ มีฟังก์ชันอื่นๆ ที่ใช้งานเป็นประจำ ดังนี้

- ฟังก์ชัน **Awake()** และฟังก์ชัน **Start()**

Awake() : References between scripts, initialization

Start(): Once scripts component is enables

ฟังก์ชัน **Awake()** และฟังก์ชัน **Start()**เป็นฟังก์ชันเริ่มต้นของโปรแกรม **unity** โดยมีข้อแตกต่าง คือ ฟังก์ชัน **Awake** เป็นการเตรียมก่อนเริ่มทำงานและรันต่อเนื่อง ส่วนฟังก์ชัน **Start** เป็นฟังก์ชันในการเริ่มทำงานและทำงานเพียงครั้งเดียว

- ฟังก์ชัน **update** ทำหน้าที่รันคำสั่งที่ประกาศตลอดเวลา เฟรมต่อเฟรม ความเร็วในแต่ละเฟรมขึ้นอยู่กับคอมพิวเตอร์ที่รัน

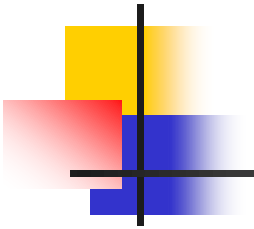
ฟังก์ชันอื่นๆ

- ฟังก์ชัน **FixedUpdate** ทำหน้าที่รันคำสั่งที่ตามเวลาที่กำหนด ใช้สำหรับงานที่ต้องการความเร็วเท่าๆ กัน

- ฟังก์ชัน **enabled** ฟังก์ชันการเปิดและปิดการใช้งาน

- ฟังก์ชัน **.active** และ **.activeself** เป็นฟังก์ชันในการตั้งค่า **active** ให้กับ **gameobjects** ทั้งการใช้ **setactive** และ **activeself**

- ฟังก์ชัน **.translate()** และ **.rotate()** เป็นฟังก์ชันในการการหมุนและการเคลื่อนที่



บทสรุป

เมื่อผู้เรียนอ่านเนื้อหาในบทนี้แล้ว สามารถเข้าใจรูปแบบ ขอบเขต การเข้าถึงฟังก์ชัน ว่ามีการใช้งานในโปรแกรมอย่างไร การประกาศใช้ฟังก์ชันทำอย่างไร เพื่อให้โปรแกรมลดการซ้ำซ้อนและง่ายต่อการทำความเข้าใจ รวมถึงฟังก์ชันทางคณิตศาสตร์ และฟังก์ชันเกี่ยวกับเวลา ที่เป็นฟังก์ชันที่มีมาพร้อมกับโปรแกรม ทำให้เกิดความเข้าใจการใช้งานฟังก์ชันเพิ่มมากขึ้น และจะช่วยลดระยะเวลาในการพัฒนาโปรแกรม

ใบงาน

ให้นักศึกษาแบ่งกลุ่มละ 2-3 คน พัฒนาโปรแกรมที่เรียกใช้งานฟังก์ชันทางคณิตศาสตร์ พร้อมยกตัวอย่างประกอบ อย่างน้อย 2 ฟังก์ชัน และนำเสนอหน้าชั้นเรียน