

Final Report

Functional Programing

เนื้อหา

จากที่ได้ Sample ไปเมื่อวันที่ 11/3 จะเห็นได้ว่า C# ได้มีการนำเอา Functional Programming เข้ามาประยุกต์ใช้มากขึ้น ซึ่งจากตัวอย่างเราสามารถสร้างตัวแปรที่ชื่อว่า Func ที่เก็บค่า function เอาไว้ได้โดยตัวอย่างจะแสดง function ที่คำนวณค่าเวลาเพื่อใช้ในการแสดงผล UI เป็นวินาที จะเห็นได้ว่าเราจะประกาศตัวแปรที่ชื่อว่า Func โดยมี parameter คือ <float,float,float> ซึ่งคล้ายกับการประกาศ type ในภาษา functional เช่น Haskell (รับ Input เป็น float 2 ตัว แล้ว output เป็น float หรือก็คือ a -> a -> a)

```
System.Func<float,float,float> cal_time = delegate(float from,float now) {return from - now;;};
```

จากตัวอย่าง นี่คือ function ที่คำนวณเวลาโดยแสดงผลออกเป็นวินาทีแบบปกติทั่วไปของภาษา c#

```
public float calculate_time(float startTime){  
    float x = Time.time;  
    return (x - startTime);  
}
```

ซึ่งเราไม่ต้องประกาศเป็น function แยกเหมือนตัวอย่างข้างบนแล้ว สามารถสร้างตัวแปรที่เป็น function ใน บรรทัดนั้นๆได้เลย

```
if(Wait_Win.win_or_not == true) Finnish();  
System.Func<float,float,float> cal_time = delegate(float from,float now) {return from - now;;};  
string minutes = ((int) cal_time(Time.time,startTime) / 60).ToString();  
string seconds = (cal_time(Time.time,startTime) % 60).ToString("f1");  
if(seconds.Length == 3) seconds = "0" + seconds;  
timerText.text = minutes + ":" + seconds;
```

Project ที่ผู้พัฒนาได้เลือกมาทำ functional programming คือ project เดิม (เกม Escape) โดยที่ผู้พัฒนาได้หยิบยก feature ใน to do list ซึ่งอันนี้ได้มีการ pospond ไปหลายรอบมากๆ กับ feature หลักๆคือ ผู้พัฒนาคิดวิธีในการ implement ไม่ออก มันคือ การสร้าง map เกมด้วยการ Input file .txt เข้าไป โดยใช้หลักการของ functional programming ผ่าน classlib ที่ชื่อ Option หรือถ้าในภาษา Haskell ก็คือ Maybe นั่นเอง

Optional

จากการที่ได้เรียนรู้ในการใช้ตัว option นี้คือ ได้เรียนรู้การสร้าง library .net ของตัวเองรวมไปถึงการ import เข้าไปใน unity ผ่านไฟล์ .dll ซึ่งเป็นองค์ความรู้ใหม่ที่ผู้พัฒนาได้ลองทำเลยทำให้ใช้เวลาพอสมควรในการศึกษาข้อมูลในการ

import โดยปกติแล้วใน Unity พวก package ต่างๆ สามารถ download และ import มาได้เลยใน package manager ของ unity แต่ว่าถ้าเมื่อเราจะใช้ package ที่เป็นของตัวเองล่ะ?(class option) ผู้พัฒนาเลยได้สืบค้นวิธีการเพื่อสามารถนำ classlib ของตัวเองไปใช้ในตัว unity ผ่านไฟล์ .dll

ซึ่งในคลาส option จะมีอยู่ 2 ตัวแปรหลักๆ คือ none กับ some เลยทำให้ผู้พัฒนาได้นำเอา pattern matching ซึ่งในที่นี้คือ method ที่ชื่อว่า match ซึ่งจากตัวอย่างคือ สร้างตัวแปรที่ชื่อว่า someblock และใช้ method SomeWhen เมื่อ asset.text.Length มากกว่า 2 จะทำให้ someblock มีค่าเป็น some ที่มีขนาดของ asset.text.length อยู่ มีเช่นนั้นเป็น none และหลังจากนั้นใช้ someblock.matchnone คือถ้า someblock เป็น none จะไปเรียก function wrongInput

```
var someBlock = (asset.text.Length).SomeWhen(s => s > 2);
someBlock.MatchNone(() => {
    wrongInput(-1);
});
```

และผู้ใช้ก็ได้มีการใช้หลักการประมาณนี้ซึ่งคล้ายๆ กันในการสร้าง feature

```
for(int i = 0 ; i < row ; i++){
    for(int j = 0 ; j < col ; j ++){
        option = map_form[i][j].SomeWhen(x => int.Parse(x.ToString()) <= 5);
        option.Match(
            some: b => create_block(int.Parse(map_form[i][j].ToString()),i,j),
            none: () => wrongInput((float)i)
        );
    }
}
```

และนี่คือ code ทั้งหมด

```
TextAsset asset = Resources.Load("stage1") as TextAsset;
var someBlock = (asset.text.Length).SomeWhen(s => s > 2);
someBlock.MatchNone(() => {
    wrongInput(-1);
});
string[] fLines = Regex.Split ( asset.text, "\n|\r|\r\n" );
Build_it(fLines,fLines.Length,fLines[0].Length);
```

```

void Build_it(string[] map_form, int row , int col){
    var option = new Option<char>();
    for(int i = 0 ; i < row ; i++){
        for(int j = 0 ; j < col ; j ++){
            option = map_form[i][j].SomeWhen(x => int.Parse(x.ToString()) <= 5);
            option.Match(
                some: b => create_block(int.Parse(map_form[i][j].ToString()),i,j),
                none: () => wrongInput((float)i)
            );
        }
    }
}

void wrongInput(float i){
    float row = i;
    var option = row.NoneWhen(a => a == -1);
    option.Match(
        some: b => print($"Check Your Input Format At Row {b + 1}."),
        none: () => print("Impossible!!!")
    );
    print("here");
    Application.Quit();
}


void create_block(float b,int i,int j){
    float num = b;
    var option = num.NoneWhen(a => a == 0);
    option.MatchSome(a => {
        Instantiate(Resources.Load($"Blocks/{a}"), new Vector3(i * 2, 0, j * 2), Quaternion.identity);
    });
}

```

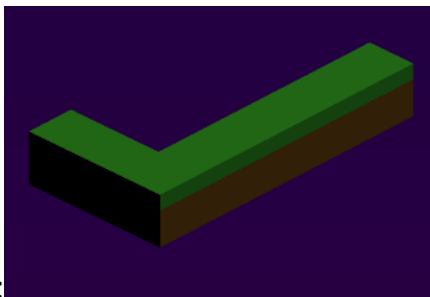
ในตัว code เองอาจจะดูน้อยใช้ครบมันน้อย แต่ว่าใช้เวลาคิดเยอะมากกกกก โดยปกติแล้วถ้าเป็นแนวนี้ผู้พัฒนาจับใส่ if else หมดแล้วมันจะยุ่งตุงนังไปหมด ยกตัวอย่าง code

```
foreach(JToken token in jt_get){
    print((string)token["name"]);
    if((string)token["name"] == "move"){
        float val = (float)token["value"];
        steps = val;
        num_block++;
        to_move = true;
        yield return new WaitWhile(() => to_move == true);
        steps = 0;
    }
    else if((string)token["name"] == "turn"){
        string val = (string)token["value"];
        if(val == "left")
        {
            degree = (float)(-90.0);
        }
        else if(val == "right")
        {
            degree = (float)90.0;
        }
        to_turn = true;
        yield return new WaitWhile(() => to_turn == true);
        degree = 0;
    }
    else if((string)token["name"] == "jump"){
        to_jump = true;
        yield return new WaitWhile(() => to_jump == true);
        steps = 1;
        to_move = true;
        yield return new WaitWhile(() => to_move == true);
        steps = 0;
    }
}
num_block++;
```

และนี่คือผลลัพธ์ของตัว code

Input: 

Output:



สรุป

จากที่ได้ลองใช้ตัว **option** มาทำให้รู้ได้ว่า เราจะไม่ปวดหัวกับการทำ **if else** ซ้อน ๆ จากการ **implement code** ข้างต้น แต่ผู้พัฒนาคิดว่ามันอาจจะทำได้ดีกว่านี้ **clean** ตัว **code** ได้ดีกว่านี้ แต่โดยที่ผู้พัฒนาเพิ่งได้มาเจอ **classlib** นี้รวมไปถึงทุ่มเทเวลาให้กับการ **import classlib** ไปใช้เองใน **unity** ทำให้อาจจะไม่มีเวลาได้ศึกษาตัว **classlib option** นี้ทั้งหมดแต่ก็ได้เปิดโลกในการเขียน **code** ของผู้พัฒนาในอีกซีกหนึ่ง โดย **project** นี้เป็น **project** ที่ต่อเนื่องและเป็น **project** จบของผู้พัฒนาเองดังนั้น ผู้พัฒนาเล็งเห็นว่าในอนาคตจะมีการ **clean code** โดยการใช้ **ofunctional programming** เข้ามาเพื่อลดความซับซ้อนในการอ่านของแต่ละ **feature** ที่ได้ **implement** ไป หรือที่กำลังจะ **implement** ในอนาคต