

Division Module Overview

In this page we will discuss what can division system work and how can we extend and reuse it.

People included

- Developers (Especially the ones who knows how to program)

Purpose (จุดประสงค์)

Structure (โครงสร้าง)

BP_DivisionPresenter

Purpose (จุดประสงค์)

BP_DivisionModel

Purpose

BP_DivisionView

Purpose (จุดประสงค์)

Known bugs (บัคที่รู้)

Stack buffer overflow

Purpose (จุดประสงค์)

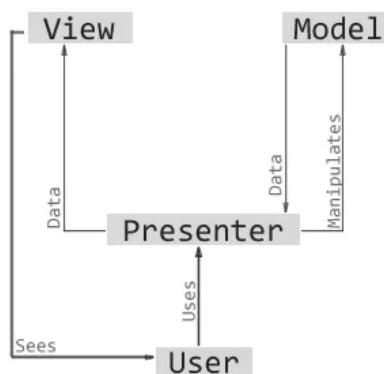
The module was designed to be mainly reusable as much as possible with ability to easy setup for any other projects.

ระบบนี้ถูกออกแบบเพื่อให้สามารถใช้ซ้ำได้มากที่สุดเป็นหลัก เพื่อให้คนที่จะนำไปใช้ต่อสามารถประกอบเข้ากับโปรเจ็คได้

Structure (โครงสร้าง)

This module is structured using [MVP Architecture](#) pattern. While it does not exactly follow architecture due to limitation in Unreal Engine context, keep in mind that it started from that architecture.

ระบบวางโครงโดยมีแบบสถาปัตยกรรมเป็น MVP ถึงแม้ว่าตัวระบบจะไม่ได้ตามสถาปัตยกรรมแบบเป๊ะๆ เพราะ Unreal Engine ก็มีสถาปัตยกรรมของตัวเอง ก็อยากให้คำนี้ไว้เสมอว่าระบบนี้เริ่มจากสถาปัตยกรรมที่กล่าวมาก่อนหน้า




 [Simple MVP Design Pattern in Swift](#)

BP_DivisionPresenter

Presenter in MVP architecture is where outside code (client) will manage. It is responsible for managing model and view as middleman. So that model and view won't need to know each other. And also, this is where client code will mostly use.

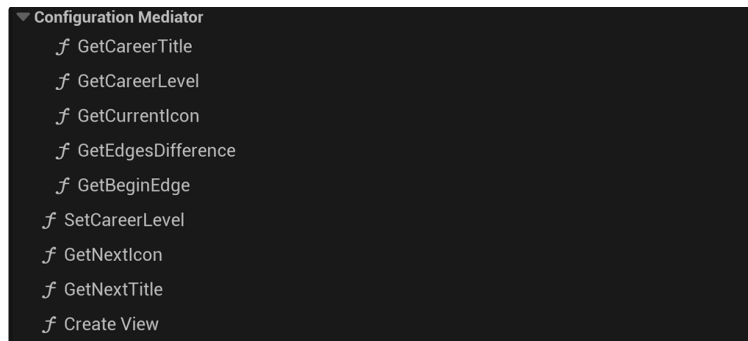
ในสถาปัตยกรรม MVP Presenter(ตัวนำเสนอ) คือส่วนโค้ดที่จะถูกโค้ดอื่นนำไปใช้ (Cilent Code) มีหน้าที่หลักในการควบคุม model และ view ในฐานะคนกลาง เพื่อที่ว่า model และ view จะไม่จำเป็นต้องรู้จักกัน ส่วน presenter คือส่วนที่ client จะเรียกใช้มากที่สุด

 Cilent code refers to code that other will use your code with. Cilent โค้ด หมายถึง โค้ดของส่วนอื่นที่จะเรียกใช้โค้ดเรา

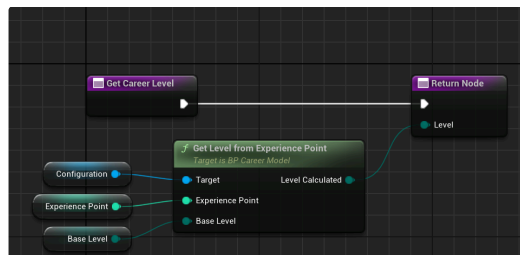
Purpose (จุดประสงค์)

Act as middle-man calling function of model and sending it to view. Also, containing some data (business) as actor has ability to mutate data in runtime environment.

ทำหน้าที่เป็นคนกลางคอยเรียกฟังก์ชันของ model แล้วส่งต่อให้ view อีกอย่างคือการเก็บข้อมูล (business) เพราะว่า actor สามารถดัดแปลงข้อมูลได้ในขณะที่ตัวเกม (environment) กำลังทำงานอยู่ (runtime)



Configuration Mediator is just function call from model. Using data contained in Presenter. Other funtion is presenter-implemented based on context.



Example of Configuration Mediator

BP_DivisionModel

Model in MVP architecture is where data (business) is contained and processed. In our context of Unreal Engine, it is Data Asset. Most of calculating logic is located here but some of data is located at presenter because Data Asset cannot do runtime-mutation.

โมเดลใน MVP คือจุดที่ข้อมูลถูกเก็บและถูกคำนวณ ในรูปแบบของ Unreal Engine นั่นก็คือ Data Asset

การคำนวณส่วนมากก็จะอยู่ที่จุดนี้ แต่ข้อมูลบางส่วนก็ต้องไปอยู่ที่ Presenter แทนเพราะว่า Data Asset ไม่สามารถแก้ไข (mutate) ข้อมูลได้เมื่อเกมกำลังทำงานอยู่ (runtime)

Purpose

To contain calculation logic and return matching output. It is where you can set up the following properties.

จุดประสงค์ เก็บตรรกะที่ใช้คำนวณ แล้วก็คืนค่าที่เหมาะสมกลับมา

LevelTitles	Byte
LevelExperience	Byte
MaximumLevel	Byte
LevelIcon	Byte

All of the function is **pure** as we do not want to change any states inside the module. We want to do that in presenter.

โค้ดทั้งหมดจะเป็น pure เพราะว่าเราไม่ต้องการที่จะเปลี่ยนเนื้อหา(State)ภายในโค้ดของเรา เราจะไปเปลี่ยนเนื้อหาใน presenter แทน

Edges
f Get Begin Edge Of Level
f Get Edges Difference
f Get End Edge Of Level
f Get Experience Point Required
f Get Titles From Level
f Get Icon From Level
f Get Level From Experience Point
f Get Experience Point From Level

BP_DivisionView

View in MVP Pattern represents where user see processed data and interacted with our presenter. As in Unreal Engine context, it is UMG.

View ใน MVP หมายถึงจุดที่ผู้ใช้จะสามารถเห็นข้อมูลที่คำนวณมาแล้วได้ และจะทำการสื่อสารกับ presenter ในบริบทของ unreal engine มันก็คือ UMG

Purpose (จุดประสงค์)

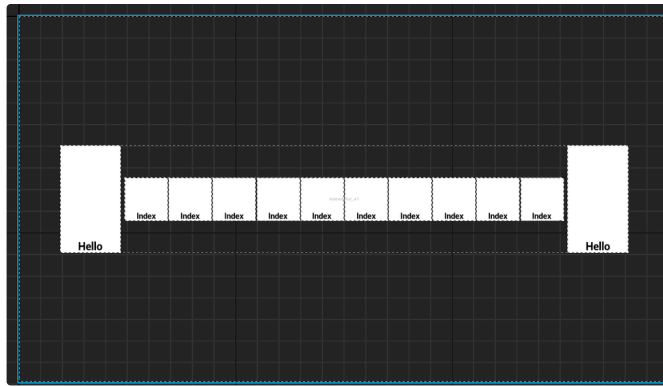
Taking user input and change presenter's state.

Default career view does not contain much of widgets. It contains only functions that can be extended/overridden by its inherited children.

รับข้อมูลขาเข้า(input) จากผู้ใช้ และเปลี่ยนเนื้อหา(state) ของ presenter

ตัว view ตั้งต้นนั้นไม่ได้มี widgets ตัวมันจะเก็บแค่ฟังก์ชันที่สามารถ ต่อทอด(extend) หรือ เขียนทับ(overridden) จากลูกที่สืบทอด(inherited children) ได้

f Update BeginEnd Icon	🔔
f Update Nodes Enabled	🔔
f Update BeginEnd Text	🔔
f Setup Minor Nodes	🔔
f Get Enabled Number Of Nodes	🔔



Known bugs (บัคที่รู้)

This is known bugs developing this module.

Please cross out fixed bug, don't remove if there is no test created.

นี่คือบัคที่รู้ระหว่างพัฒนาระบบนี้

ถ้าบัคไหนแก้แล้ว ให้ขีดฆ่าทิ้ง อย่าลบถ้ายังไม่ได้ทำการสร้างตัวทดสอบ(test)

Stack buffer overflow

This will happened when experience point is set to 1,000,000 and over. Or just input very large number (1 million+) to cause this bug.

บัคนี้จะเกิดขึ้นก็ต่อเมื่อ experience point ถูกตั้งมากกว่า 1,000,000 หรือมากกว่า หรือไม่ก็ให้ใส่ค่าเลขที่ใหญ่มากๆ เพื่อทำให้เกิดบัคนี้