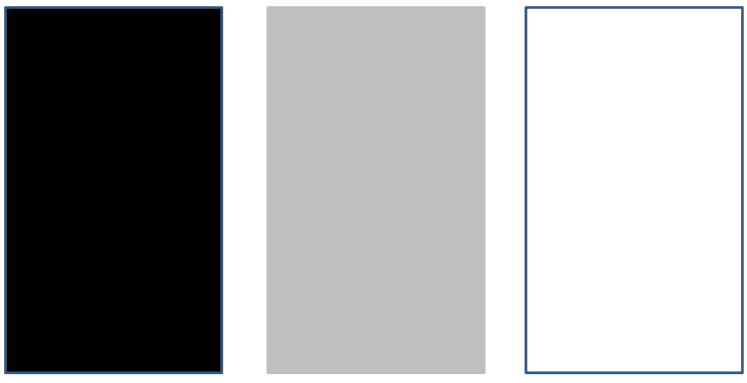
Presentation Model pattern

陳偉凱
台北科大資工系

Objective

It would be much easier, if the world is black and white only.



View Model

Objective

- An application may have several different states and behaves differently in each state
- The states must be stored and displayed

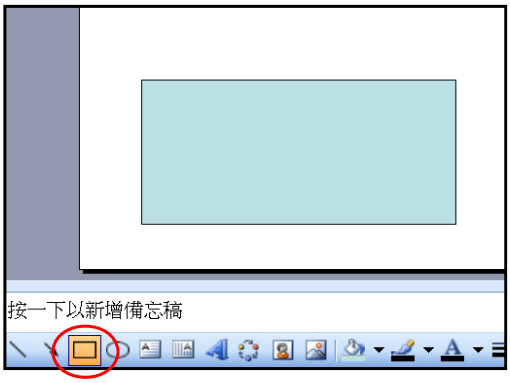
Objective

- Objective
  - Represent the state and behavior of the presentation independently of the GUI widgets used in the user interface
  - Do not store GUI states into widgets (or their properties)

所有狀態儲存在Model
View顯示狀態，而不是儲存狀態

Motivation

- GUIs consist of widgets that contain (display) the state of the GUI (or App)



Motivation

- Storing the states in widgets (e.g., Widget1.Checked)
  - makes it harder to get (from model)
  - encourages putting presentation logic in the view class.

```
void RectangleWidgetClickHandler(…)
{
    rectangleWidget.Checked = true;
    circle.Checked = false;
    …
}
```

Behaviors

Fat view

```
void MouseClickHandler(…)
{
    if (rectangleWidget.Checked == true)
    {
        // tell model to create a rectangle
        …
    }
    if (circleWidget.Checked == true)
    …
}
```

A state variable should NOT be stored in a widget

Motivation

- The states (and behaviors) of an application should be stored in a model class
  - Widget states can be derived from the states of the application

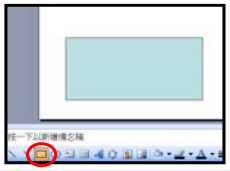
```
void RectangleWidgetClickHandler(…)
{
    model.SetDrawingMode(RECTANGLE);
    RefereshWidgetState();
}
```

Simplified

```
void MouseClickHandler(…)
{
    model.MouseClick(…);
}
```

Not a fat view anymore

```
void RefreshWidgetState(…)
{
    rectangleWidget.Checked = model.isRectangleChecked();
    circleWidget.Checked = mode.isCircleChecked();
    …
}
```

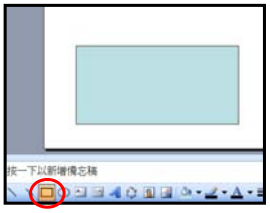


Motivation

- But, should model be responsible for implementing "isRectangleChecked()"?
  - Two different views (e.g., GUI and TextUI) may use the same model
  - Put isRectangleChecked() in model does not make sense for TextUI, which cannot (does not need to) display "checked"
  - Some view behaviors are just not suitable to put in model

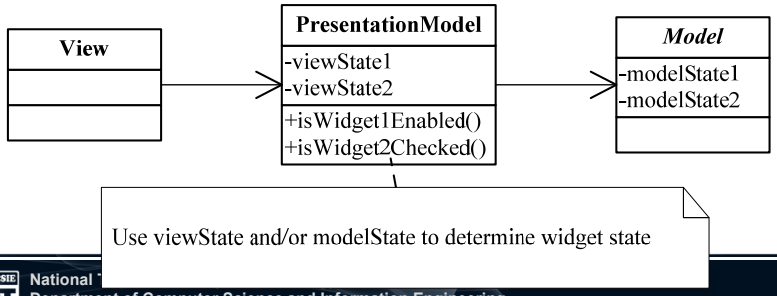
```
void RefreshWidgetState()
{
    rectangle.Checked = model.isRectangleChecked();
    circle.Checked = mode.isCircleChecked();
    …
}
```

A view-dependent behavior

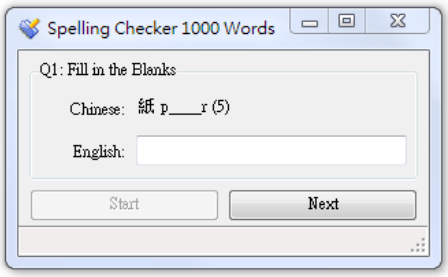


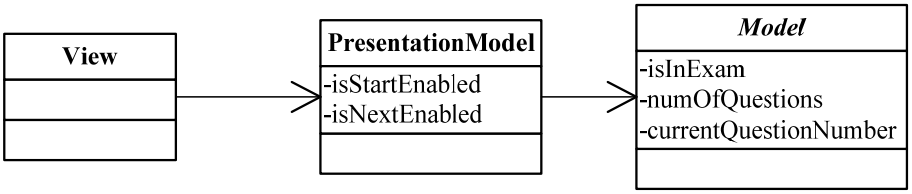
Motivation

- Presentation Model pulls the state and behavior of the view out into a model class that is part of the presentation.
  - The Presentation Model coordinates with model and provides an interface to the view that minimizes decision making in the view.
  - The view stores all its states in the Presentation Model and synchronizes with Presentation Model

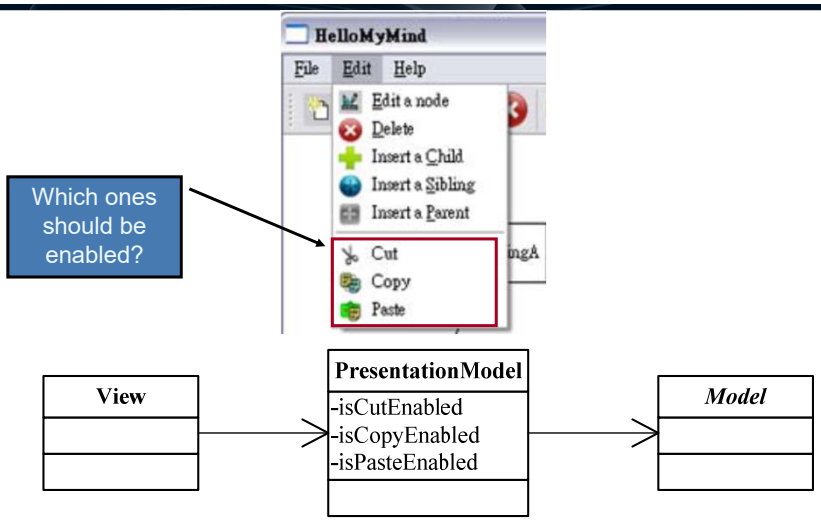


Motivation



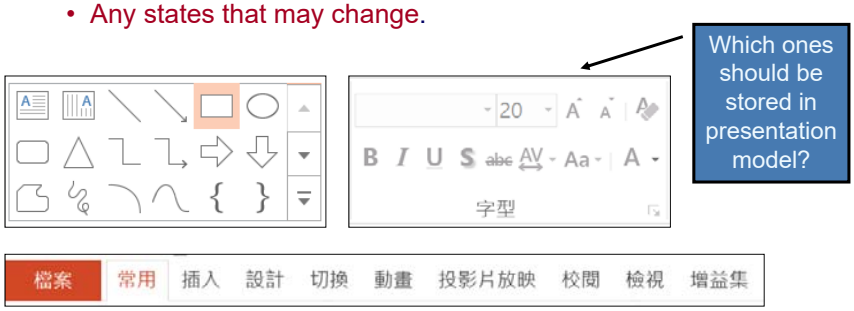


Motivation



How it works

- Presentation Model
  - Does not have any widgets
  - Maintain data for all ***dynamic information*** of the view
    - The contents of widgets and whether or not they are enabled.
    - Any states that may change.



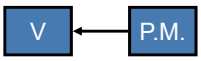
How it works

- View
  - Simply projects the state of the presentation model onto the glass.
  - All the decisions are made by the Presentation Model, leaving the view to be utterly simple.

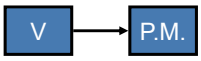
```
void RefreshWidgetState()
{
    rectangle.Checked = presentationModel.isRectangleChecked();
    circle.Checked = presentationMode.isCircleChecked();
    …
}
```


Implementation

- View ⇔ Presentation Model
  - Presentation Model references View
    - Presentation Model maintains synchronization code
    - The views implement interfaces allowing for easy stubbing when testing the Presentation Model.



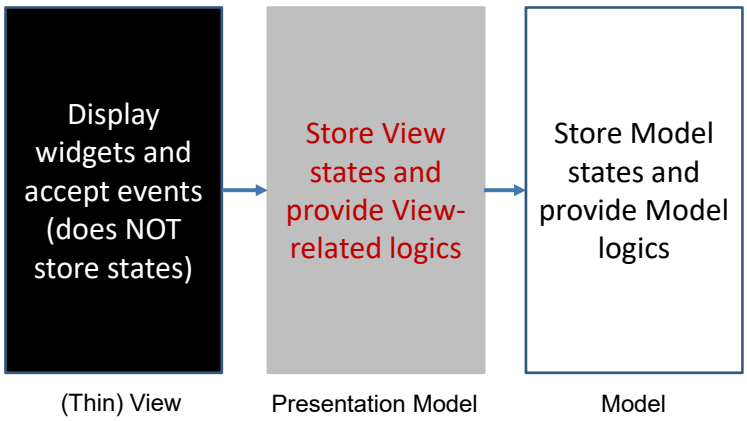
  - View references Presentation Model
    - View maintains synchronization code
    - The testing occur on the Presentation Model and not the View.



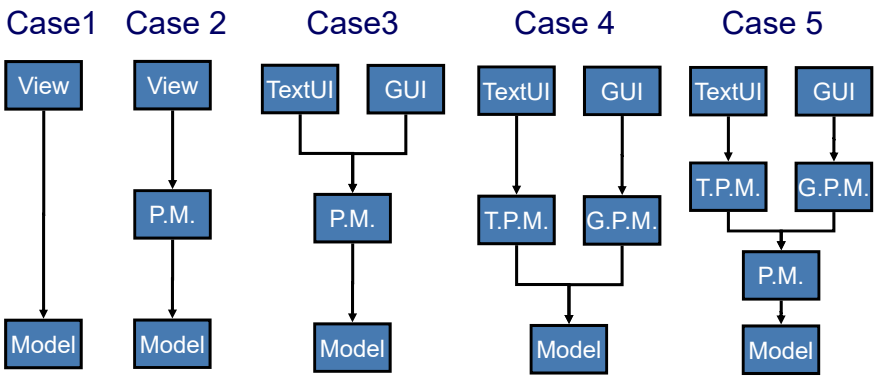
When to use it

- Presentation Model is a pattern that pulls presentation behavior from a view.
  - Allowing you to test without the UI
  - Support for multiple views and may make it easier to develop the user interface.
- Downside
  - you need a synchronization mechanism between the presentation model and the view.

How to use it

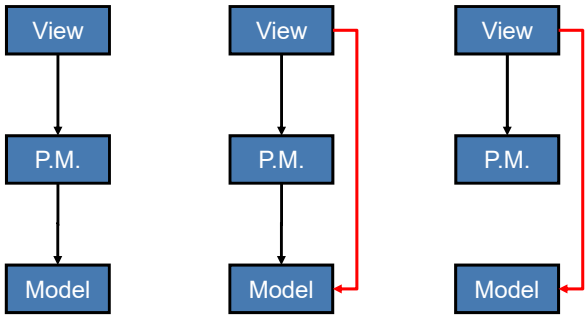


How to use it

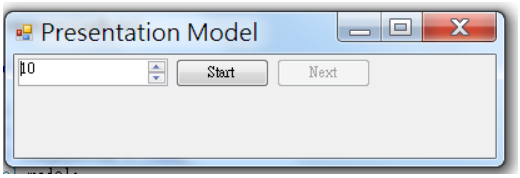


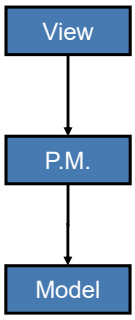
How to use it

Which of the followings are all right?



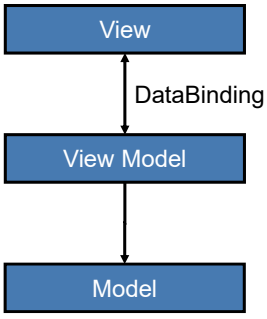
Example code





Related patterns

- Microsoft MVVM (Model View ViewModel) pattern
  - A specialization of the Presentation Model



Reference

- Martin Fowler, Presentation Model,
  http://martinfowler.com/eaaDev/PresentationModel.html