Model View Controller

BRIAN LAMARCHE

COMPUTER SCIENCE 323 – SOFTWARE DESIGN

Architectural Pattern

Model View Controller is an architectural pattern.

Intended to decouples the data from how the user sees/interacts from it.

MVC Players

Model

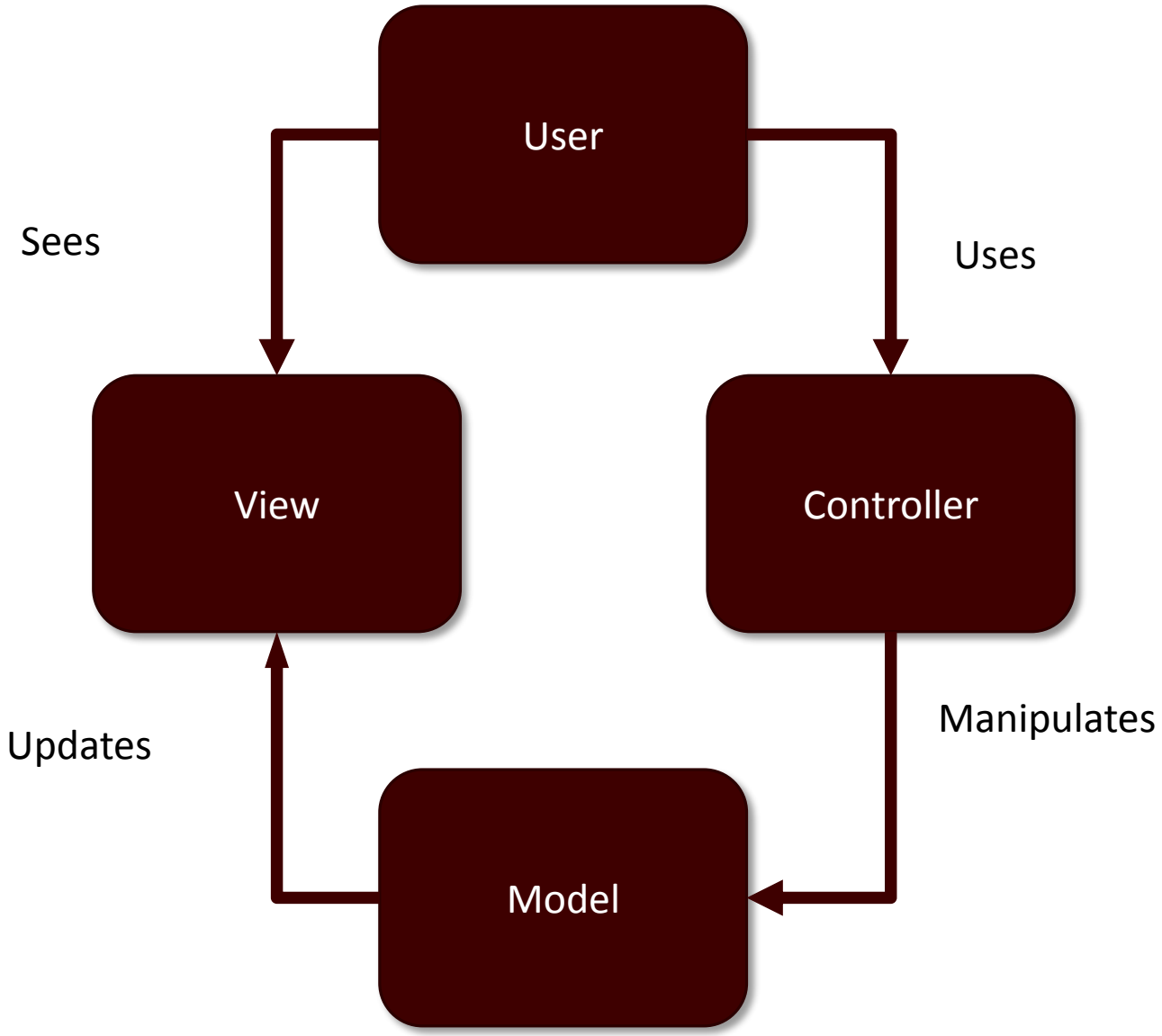
- Data
- Updates View

Controller

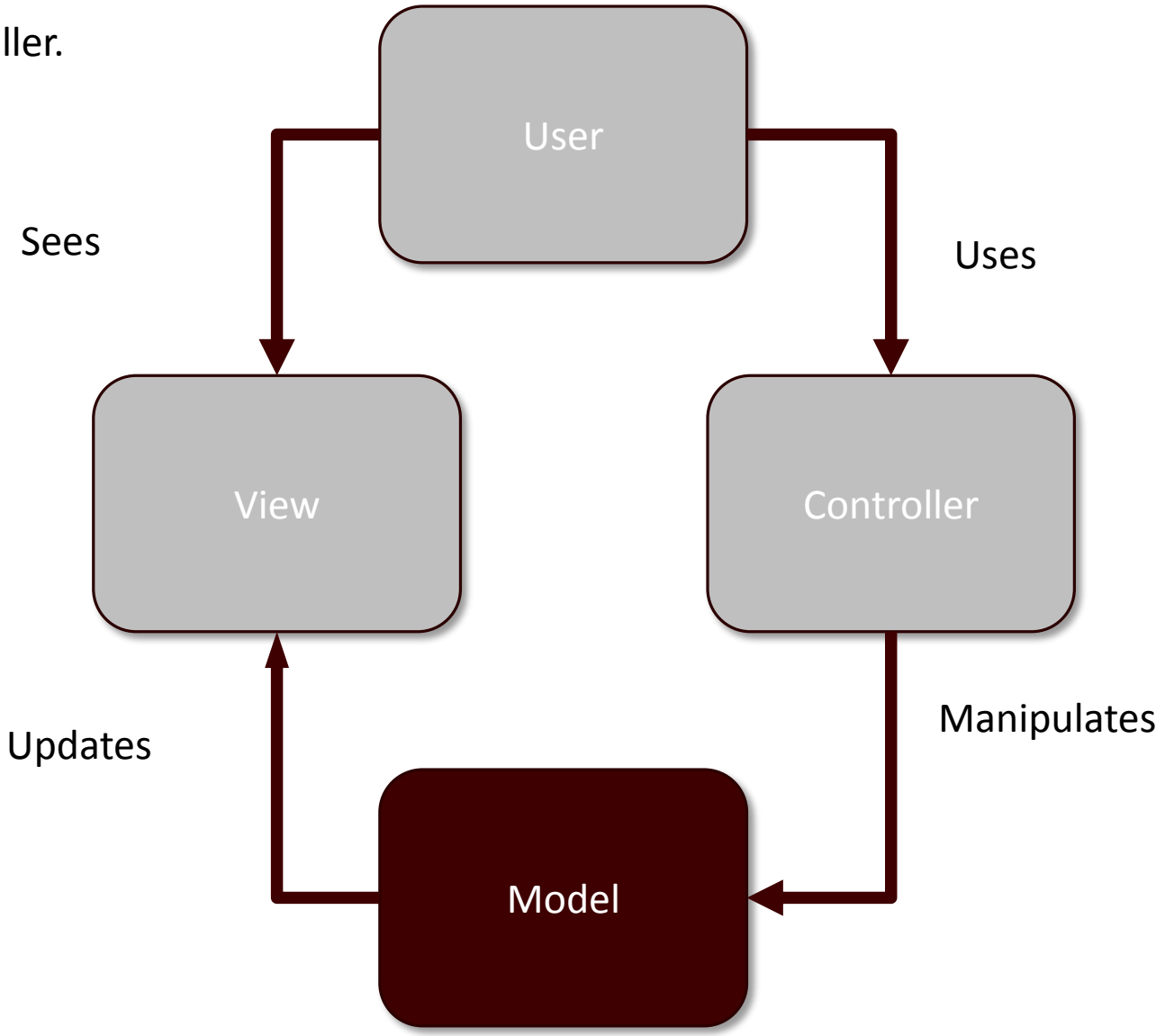
- Sends messages to view to update view’s presentation of model

View

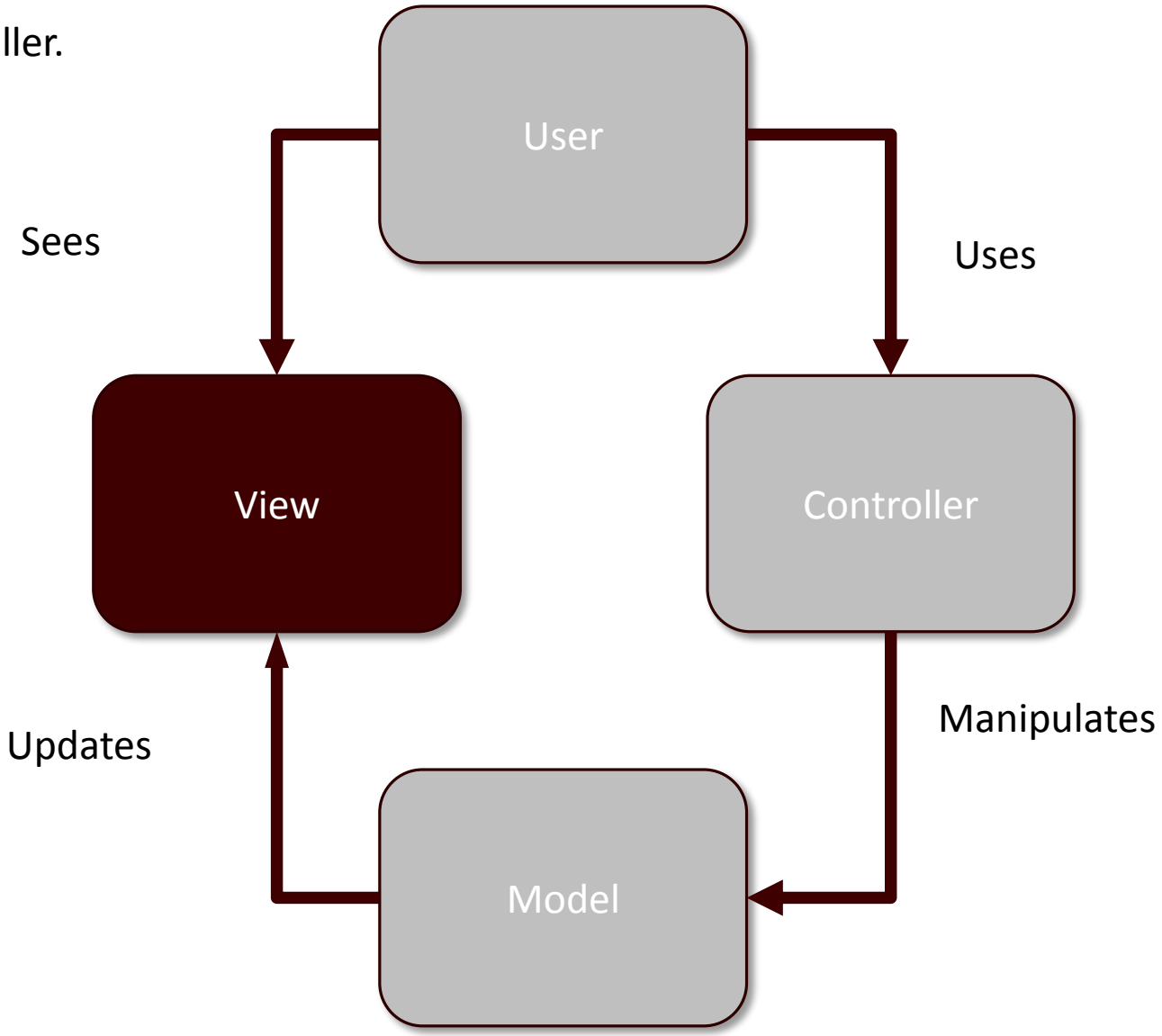
- Presents information


User
Sees
Uses
View
Controller
Updates
Manipulates
Model

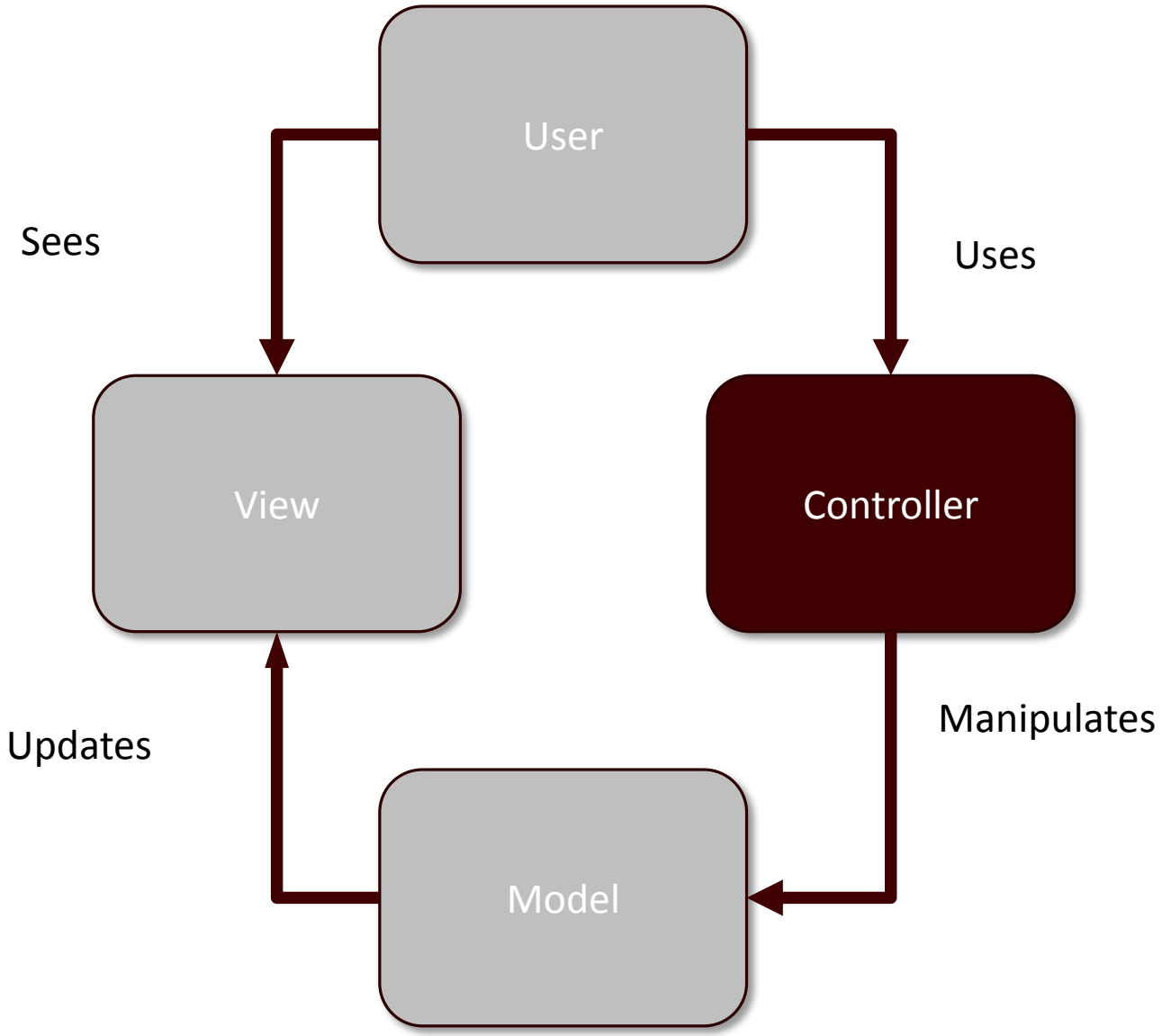
Knows nothing of the controller.
Knows nothing of the view

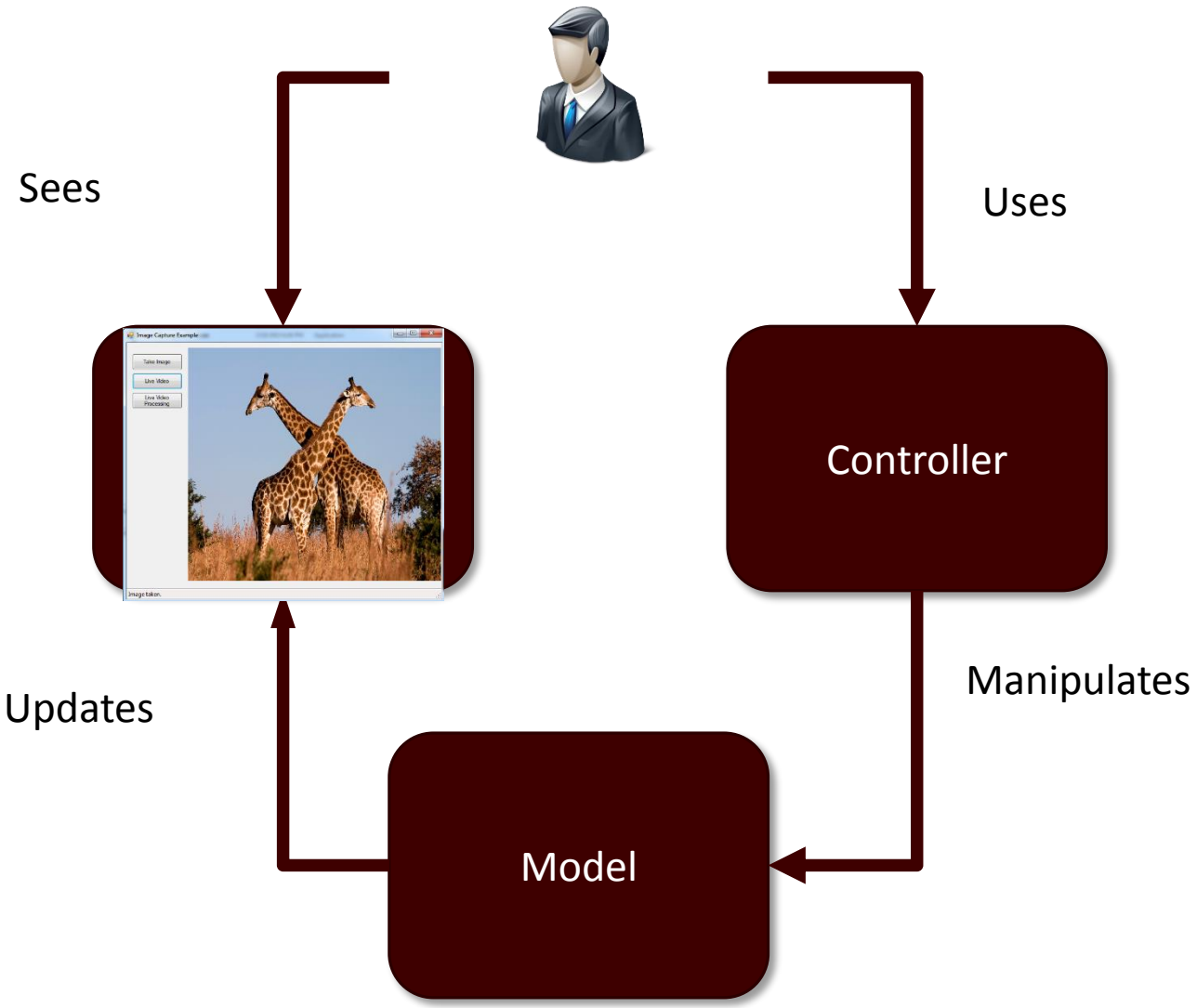


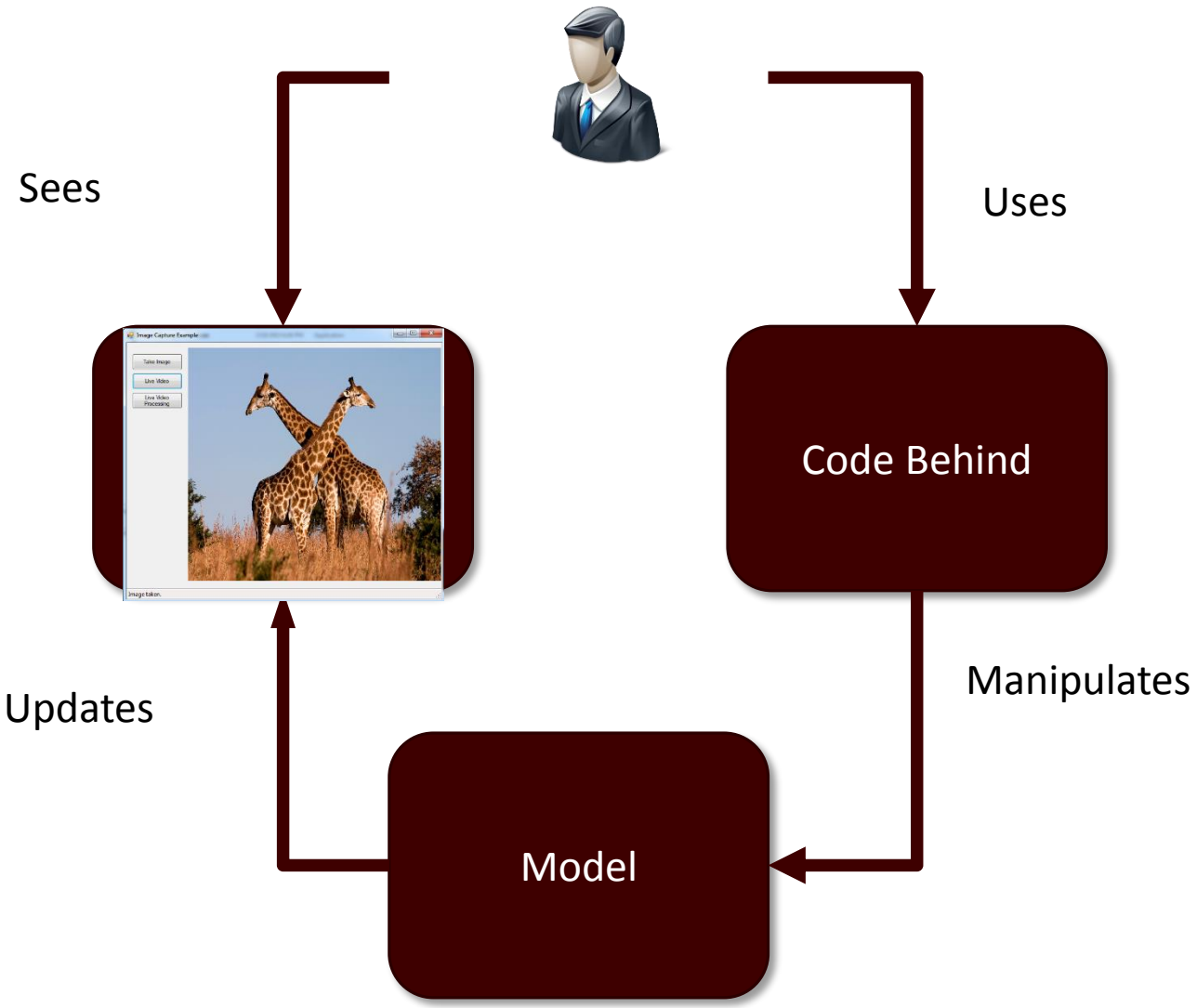
Knows nothing of the controller.
Knows about the model



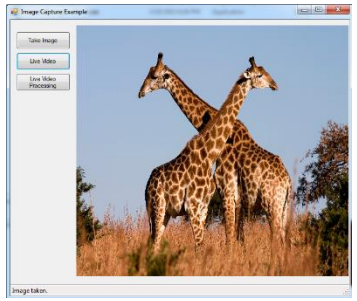
Knows about the View
Knows about the Model




Sees
Uses
Controller
Manipulates
Model
Updates


Sees
Uses
Code Behind
Manipulates
Model
Updates

Sees Uses



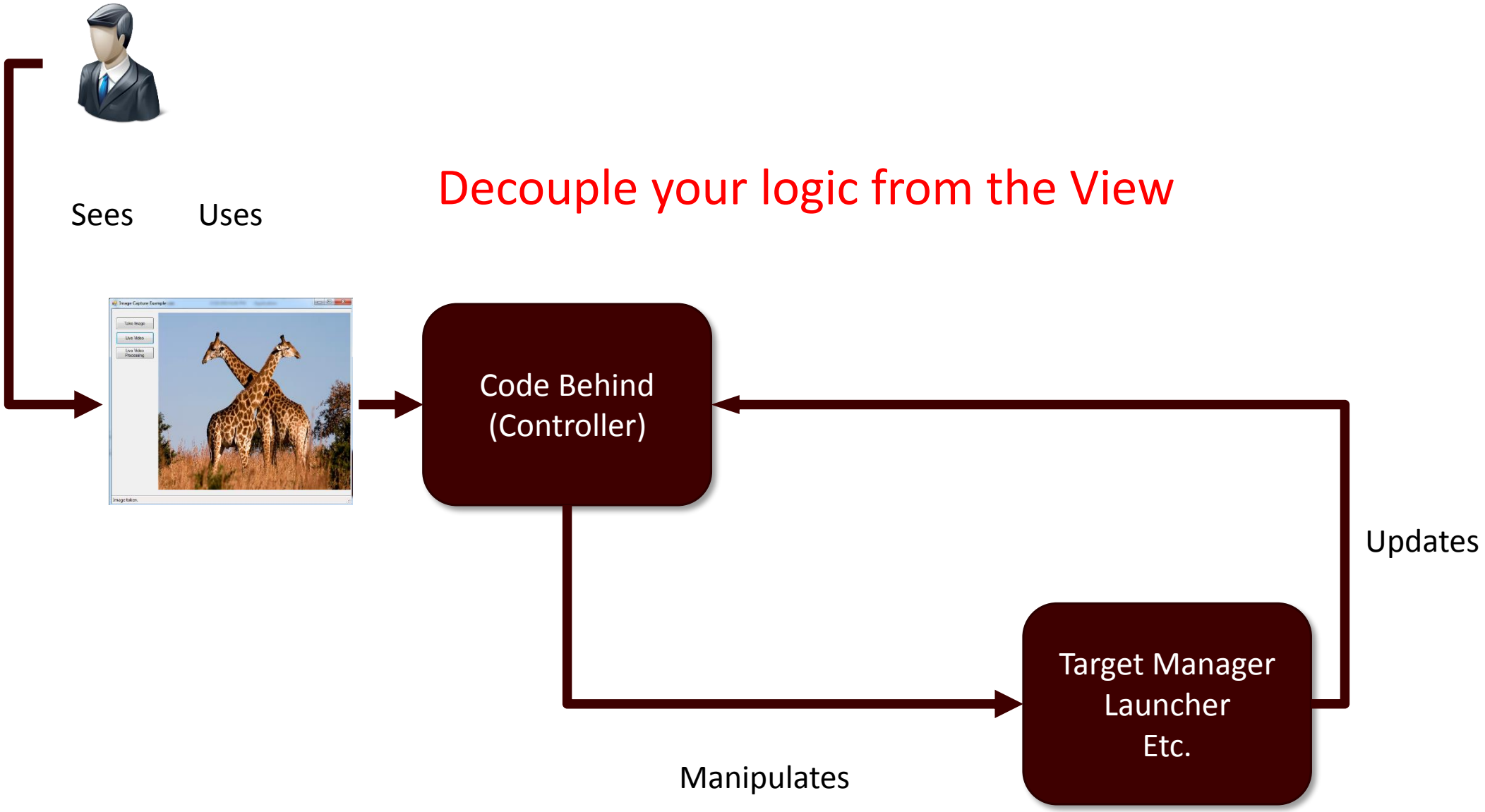
Code Behind
(Controller)

Updates

Model

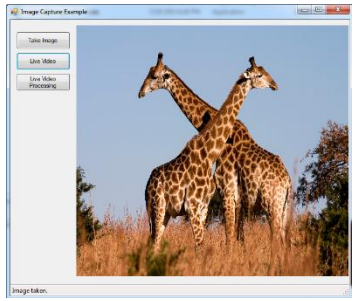
Manipulates

Decouple your logic from the View



Sees

Uses



Basic Code Behind (Controller)

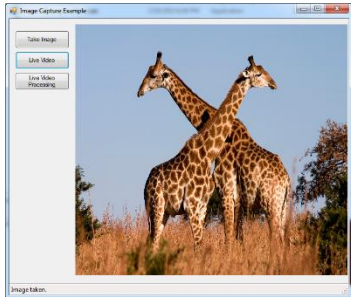
Logic (Controller)

Manipulates

Target Manager Launcher Etc.

Updates

Sees Uses



Basic Code Behind (Controller)

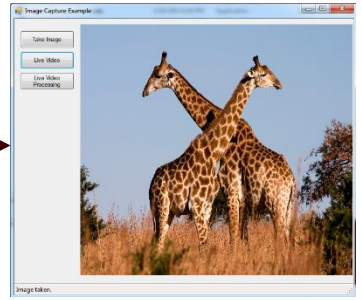
Logic (Controller)

Manipulates

Target Manager
Launcher
Etc.

Observer Pattern

Sees Uses

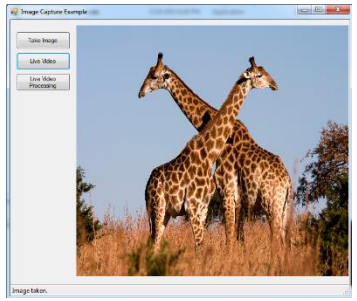


Basic Code Behind (Controller)

Logic (Controller)

Target Manager Launcher Etc.

Package



Basic Code Behind (Controller)

Package 2

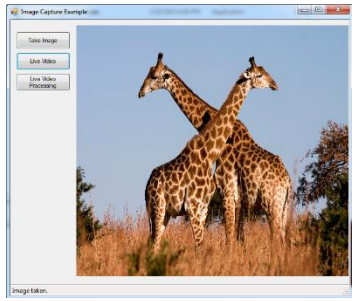
Console Application

Package 3

Logic (Controller)

Target Manager Launcher Etc.

Package



Basic Code Behind (Controller)

Package 2

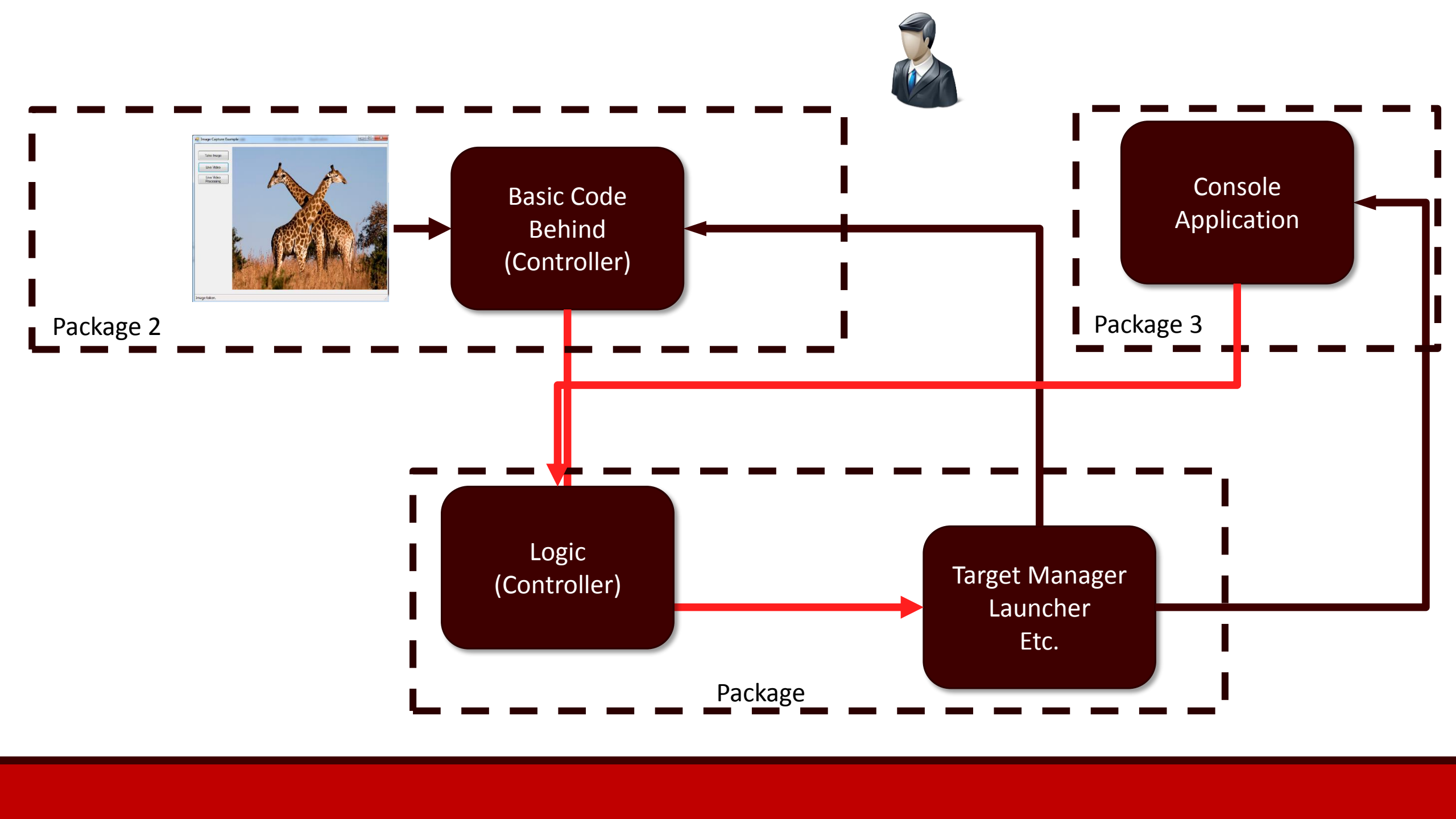
Console Application

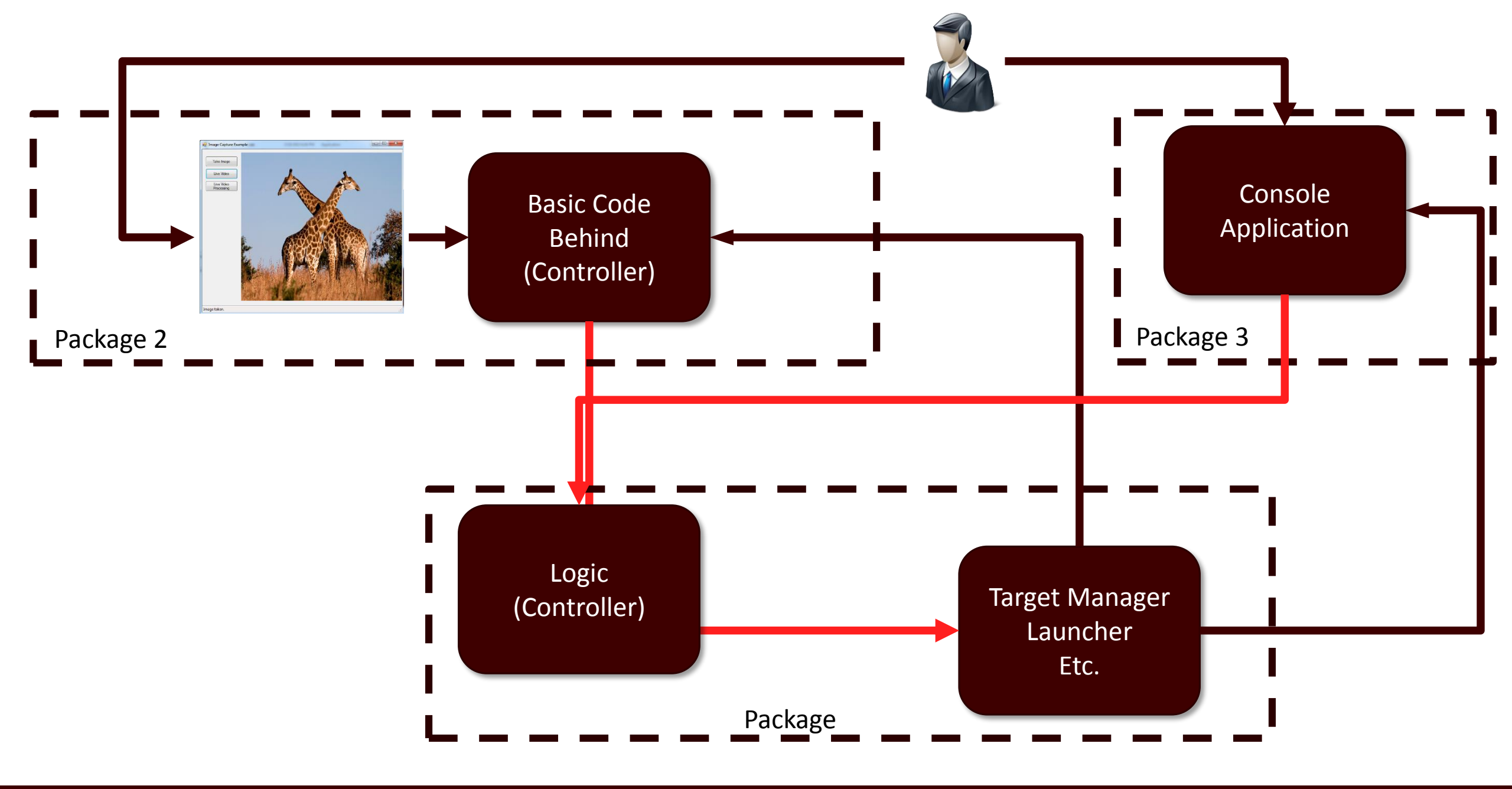
Package 3

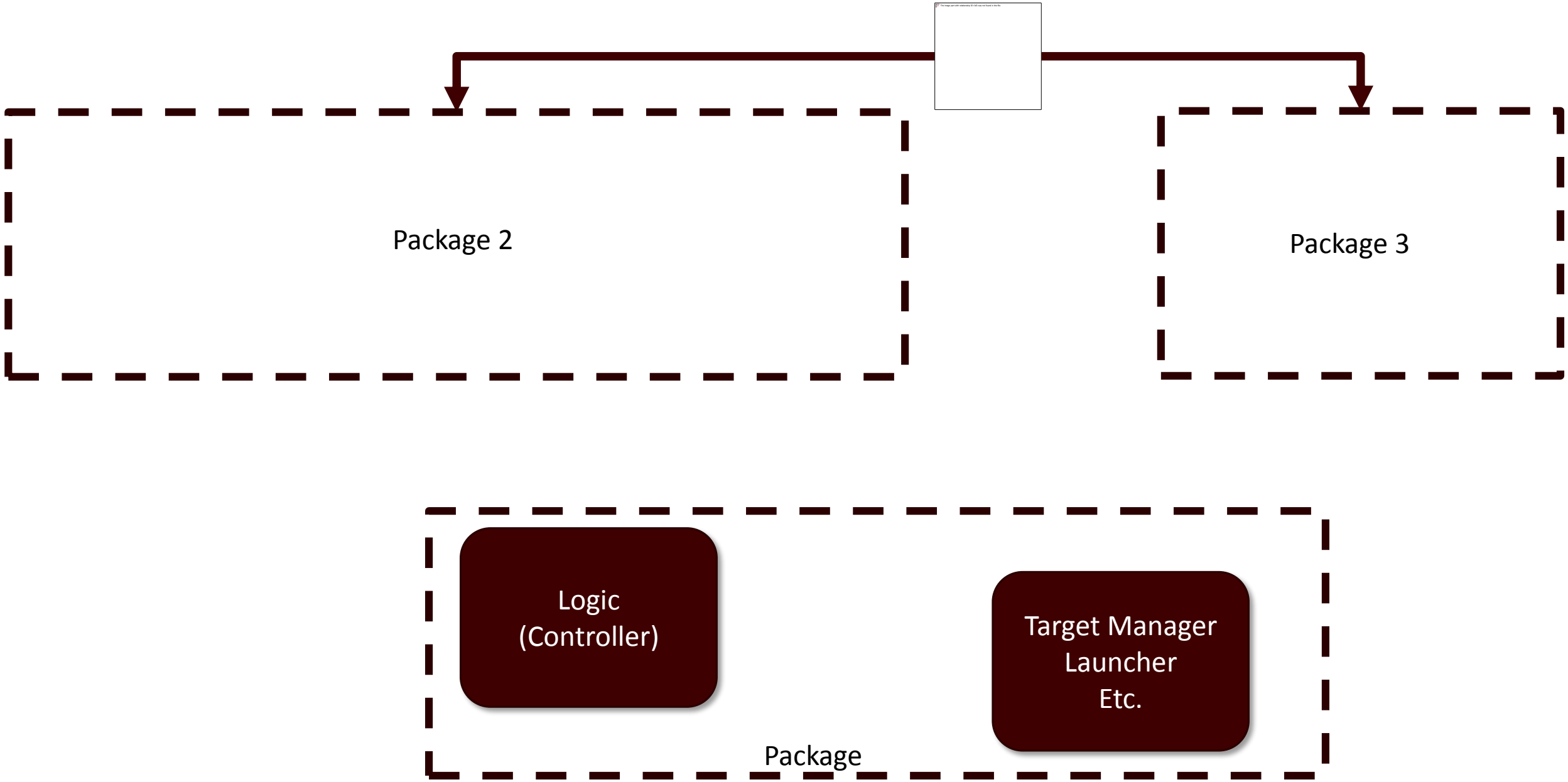
Logic (Controller)

Target Manager Launcher Etc.

Package


Basic Code Behind (Controller)
Console Application
Package 2
Package 3
Logic (Controller)
Target Manager Launcher Etc.
Package


Basic Code Behind (Controller)
Console Application
Package 2
Package 3
Logic (Controller)
Target Manager Launcher Etc.
Package


Package 2
Package 3
Logic
(Controller)
Target Manager
Launcher
Etc.
Package

Observer Pattern

Observer Pattern

The observer pattern allows for an object to notify other objects of state change.

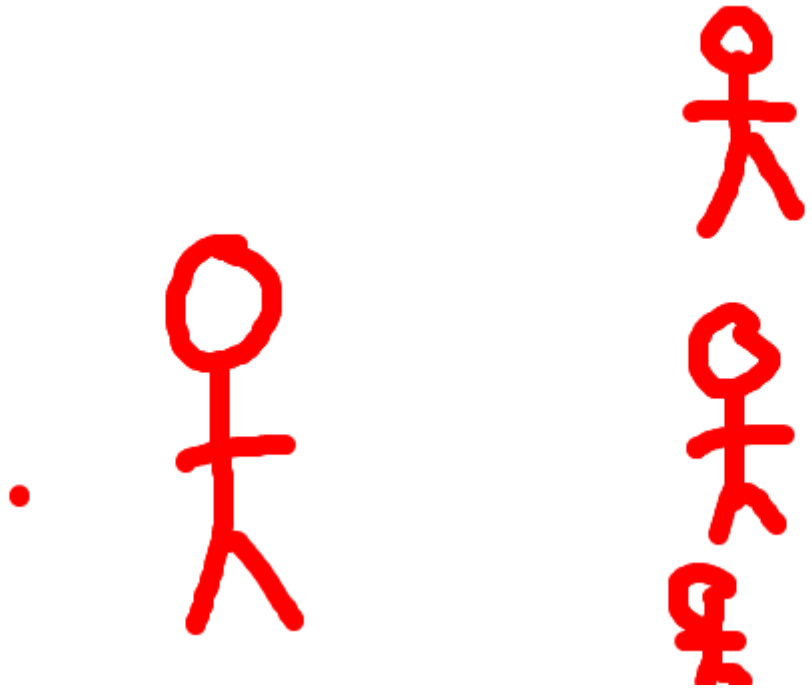
Players

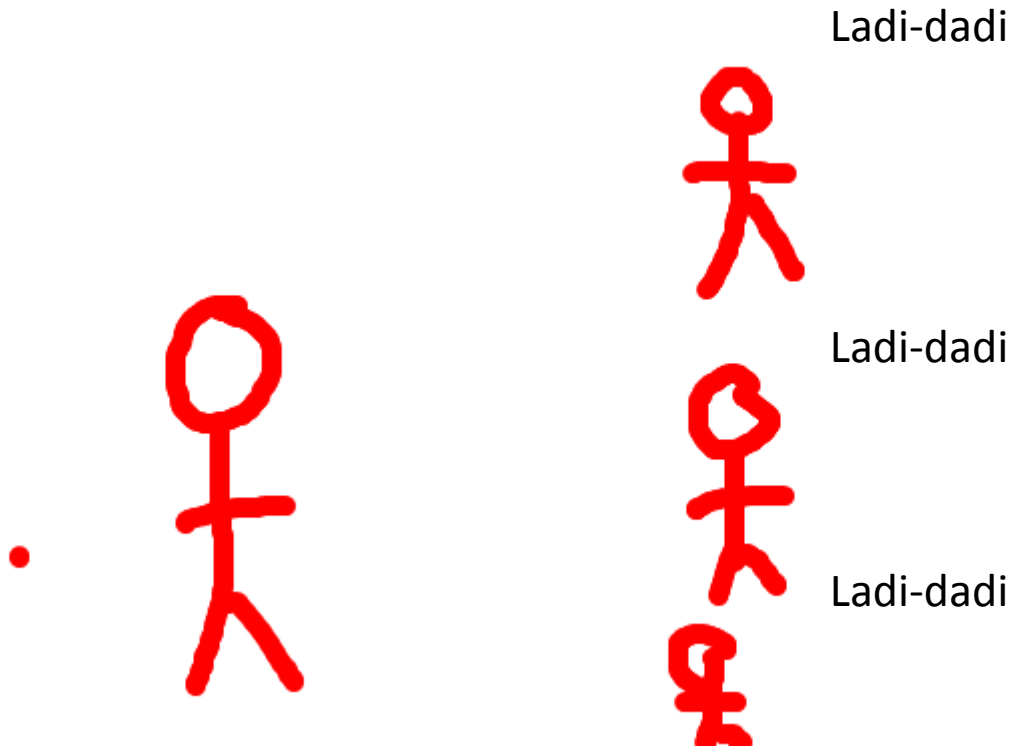
- Subject
- Observer
  - Concrete Observers

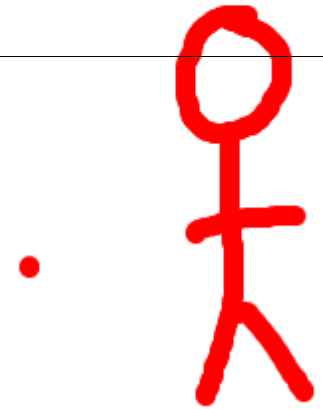
Observer Pattern

Players

- Subject
  - Manages State – Part of Model
- Observer
  - Wants notification of state change from subject
  - Model / View / Controller
  - Concrete Observers




Ladi-dadi
Ladi-dadi
Ladi-dadi



Ladi-dadi

Ladi-dadi

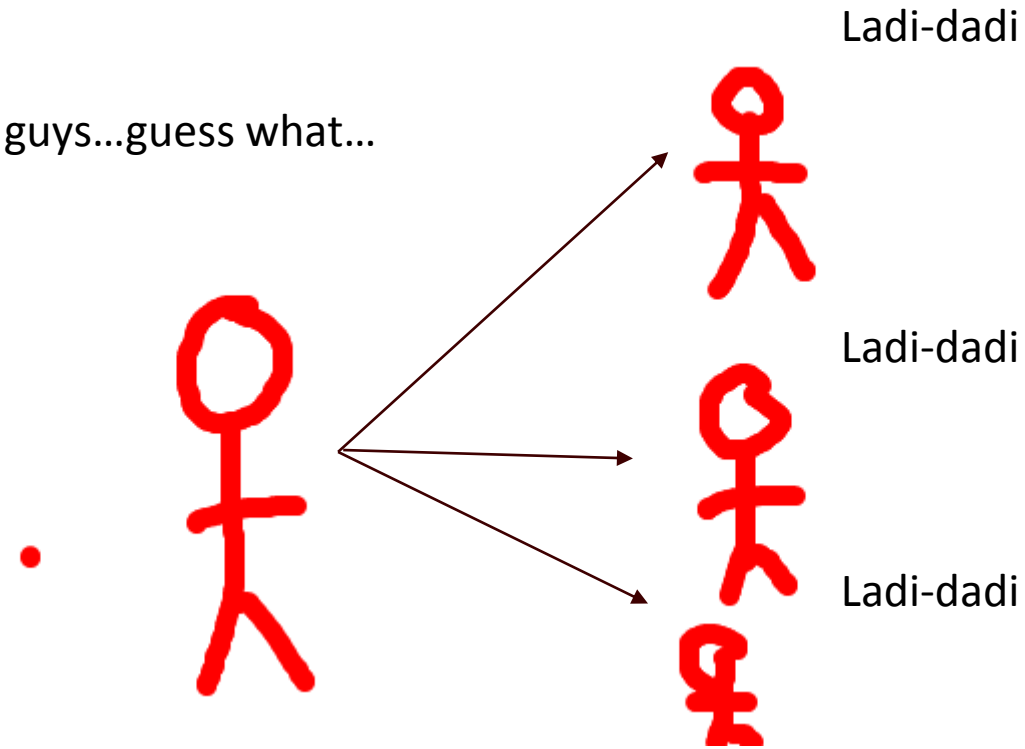
Ladi-dadi

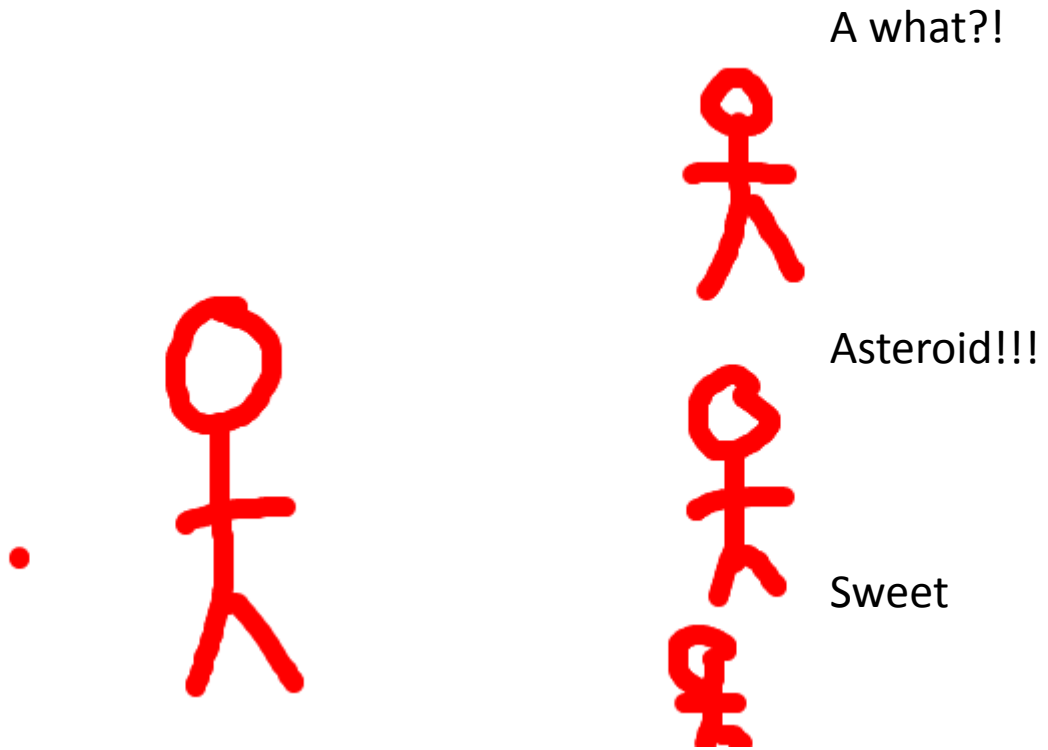
Ladi-dadi

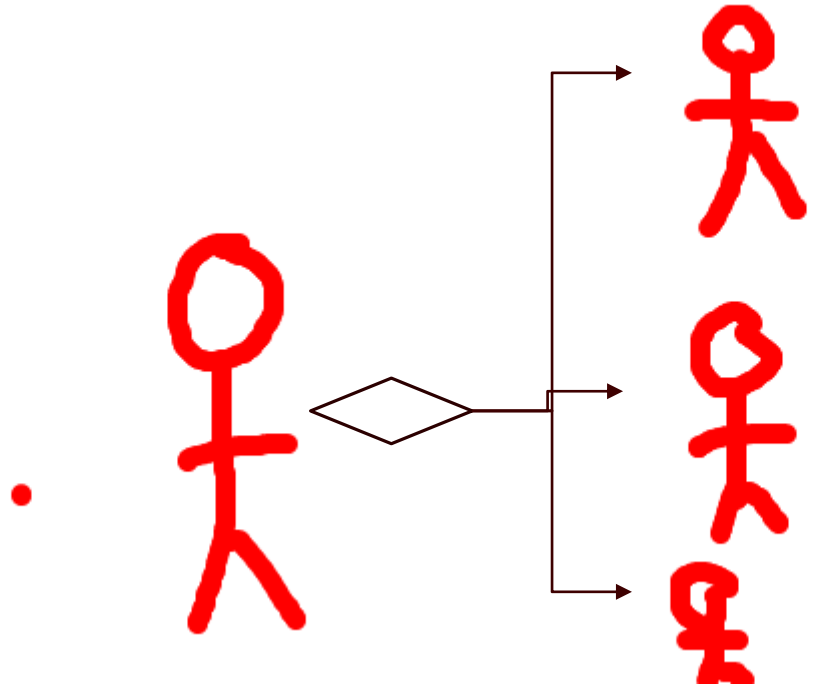
Ladi-dadi

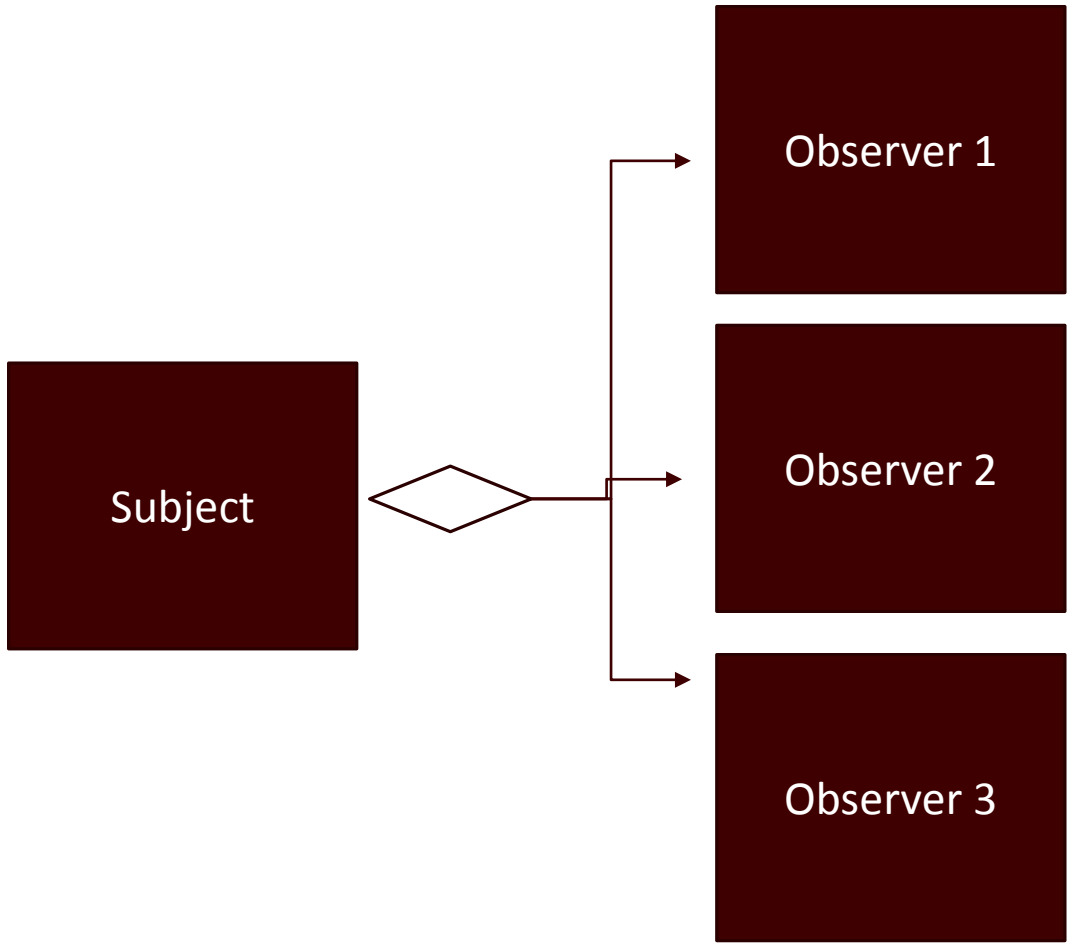
Ladi-dadi

Ughh guys…guess what…

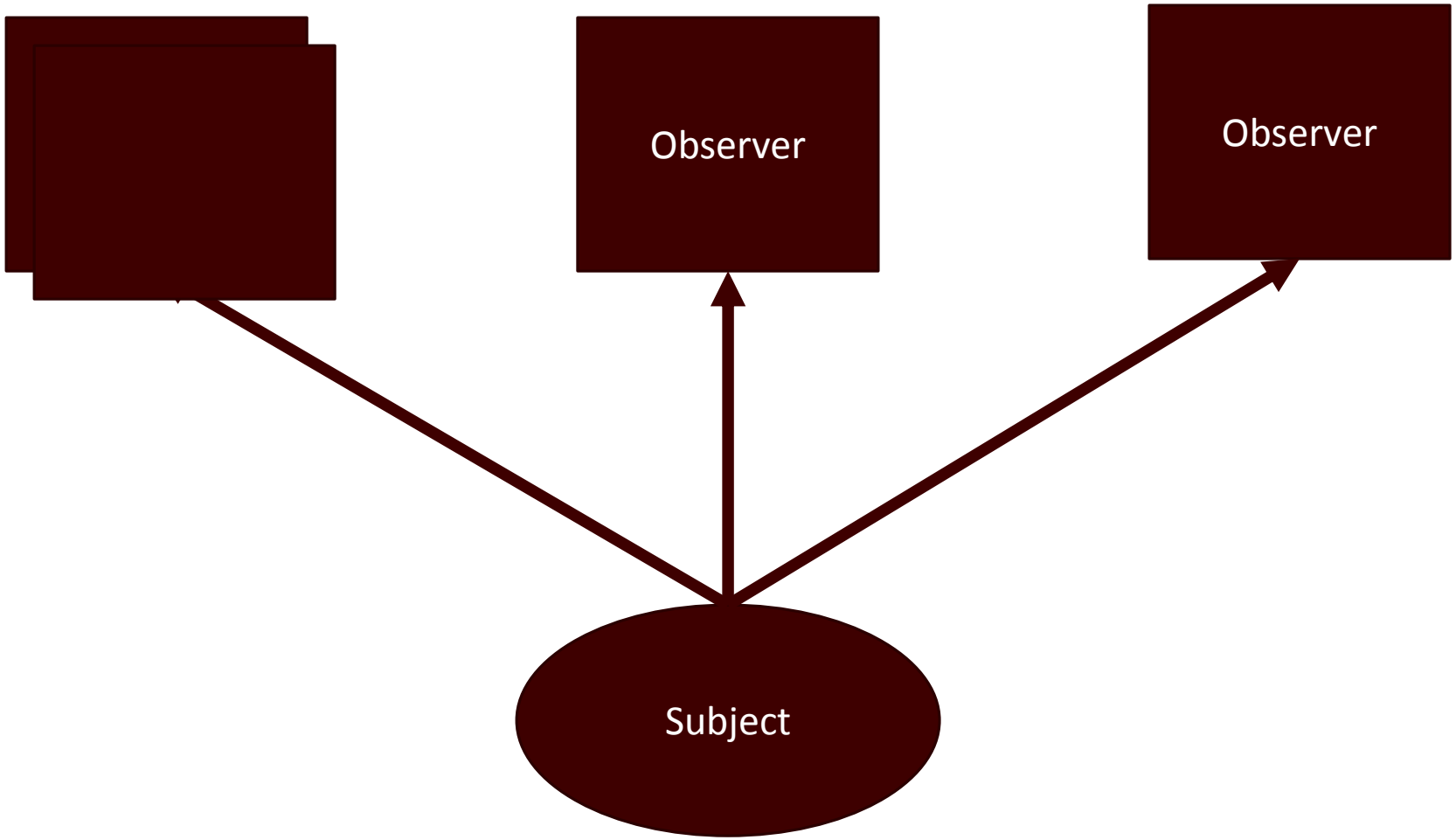



A what?!
Asteroid!!!
Sweet

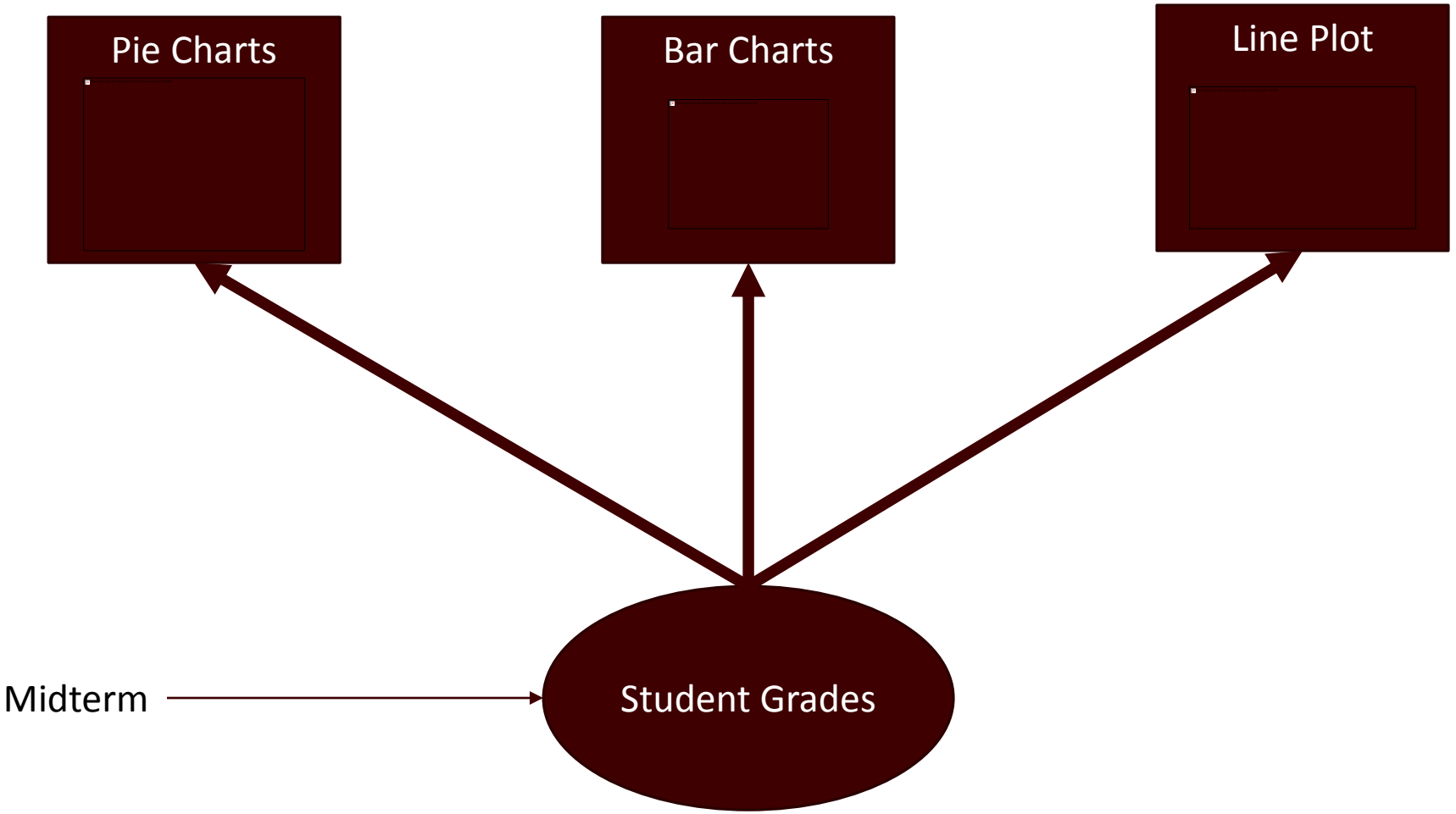



Observer 1
Subject
Observer 2
Observer 3

Or …

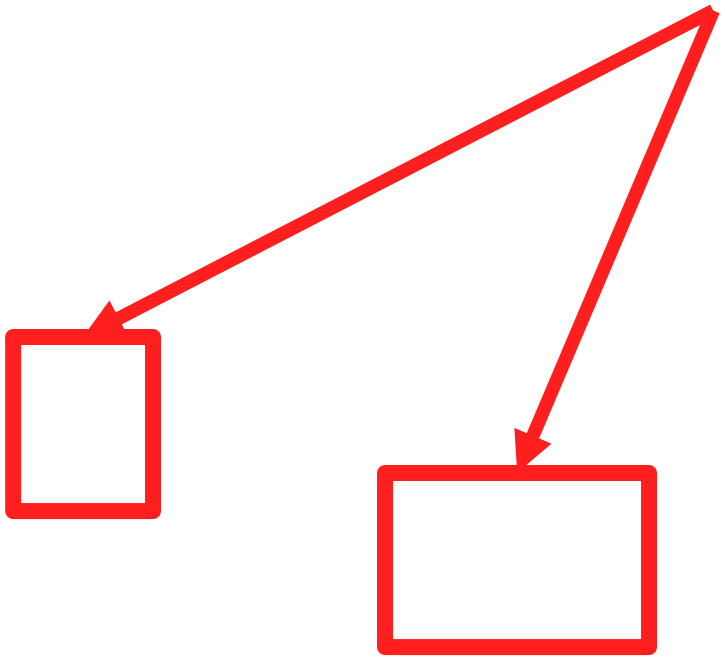


Or …

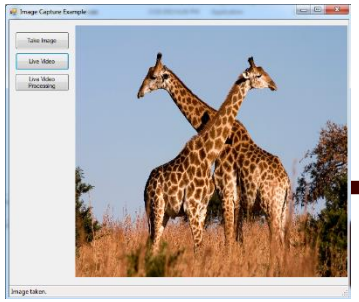


The image part with relationship ID rId2 was not found in the file.

Important Parts



Sees Uses



Basic Code Behind (Controller)

Logic (Controller)

Target Manager Launcher Etc.

Updates

Events

.net has *events*

Events are ways for classes to provide notification to observers

Multi-cast delegate

- Aggregation – multiple objects to notify!

Use:

- Delegate – e.g. function pointer, method signature
- Event Declaration

Event Tutorial: http://msdn.microsoft.com/en-us/library/aa645739(v=vs.71).aspx

Delegate

A delegate is just a method signature

```
public delegate void AddAnimal(object sender, Animal animal);
```

Delegate

A delegate is just a method signature

```
public delegate void AddAnimal(object sender, Animal animal);
```

Delegate

A delegate is just a method signature

```
public delegate void AddAnimal(object sender, Animal animal);
```

```
public event AddAnimal AddedDog;
```

Event Definition

```
public delegate void AddAnimal(object sender, Animal animal);
```

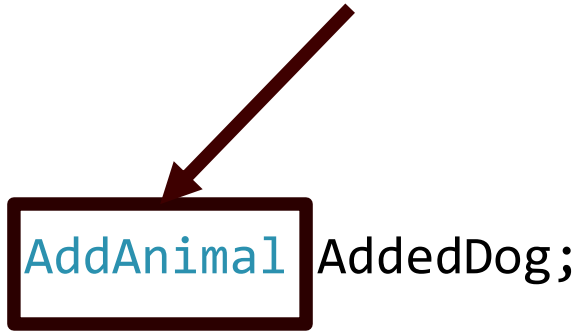
```
public event AddAnimal AddedDog;
```

Event Tutorial: http://msdn.microsoft.com/en-us/library/aa645739(v=vs.71).aspx

```
public event AddAnimal AddedDog;
```

Event Definition

```
public delegate void AddAnimal(object sender, Animal animal);
```

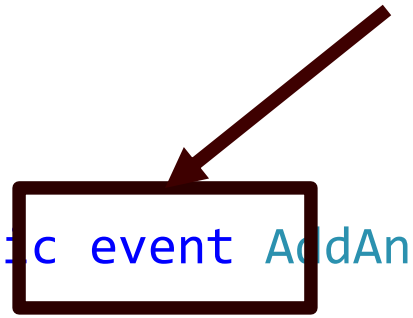


```
public event AddAnimal AddedDog;
```

```
public event AddAnimal AddedDog;
```

Event Definition

```
public delegate void AddAnimal(object sender, Animal animal);
```



```
public event AddAnimal AddedDog;
```

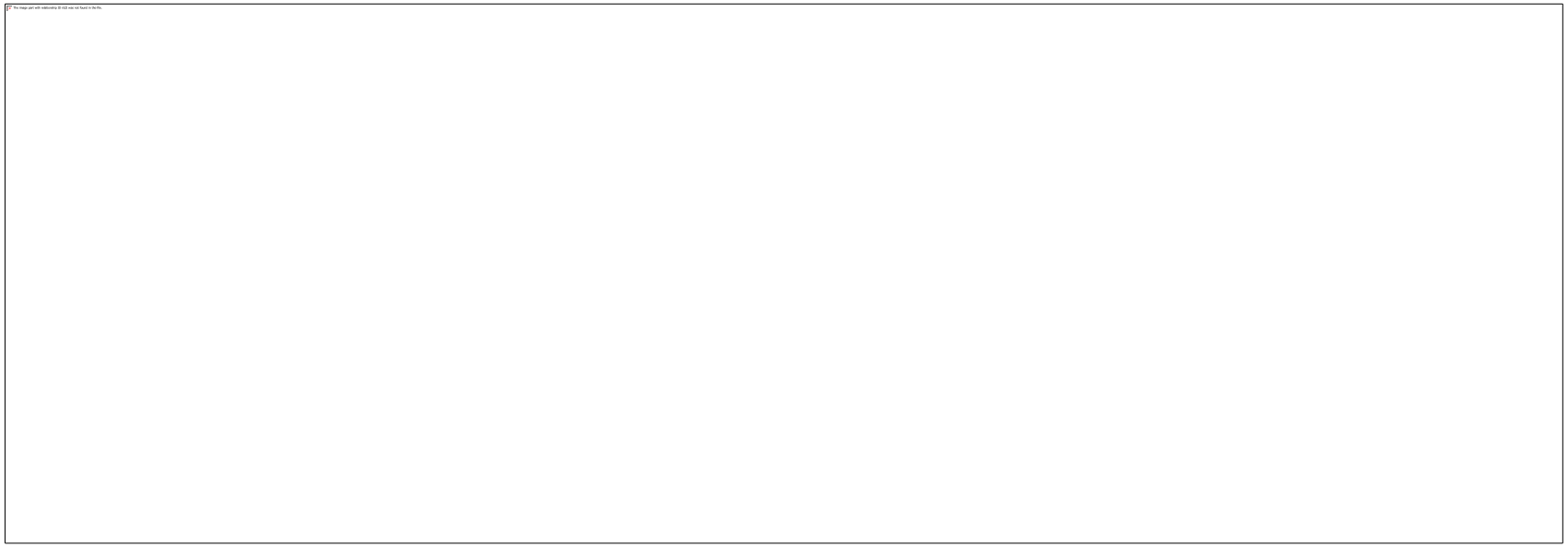
Event Subscription

```
m_manager.AddedCat += m_manager_AddedCat;
```

Event Subscription

```
m_manager.AddedCat += m_manager_AddedCat;
```

Example…



Now you go!

Implement the observer pattern using the event and delegate structure in .net

Create an example program that reads a target INI file, and displays the targets in a list. When a target list is read, it should add the target to a target manager. When a new target is added (or list of targets) the TM should notify a form and an object that logs to a file that new targets were aded