# Warzone Architecture Design



Figure - Warzone Architecture Diagram

Figure 1, shows a package diagram giving a high-level view of the Warzone application and how each package maps into the MVC (Model View Controller) architecture of the system. Details of each package is provided below.

Note: packages in the diagram contain some example classes that are both applicable to the system (but may not necessarily be realized in the final build) and that at this point simply illustrate the purpose and responsibility of each package as described below.

## Model

* Maintains and manages the state of the application, i.e. game entities such as Map, Player, Order, Country, etc…
* State change requests are defined by I\*Model based interfaces, where \* is replaced by model defined class names. State change requests are initiated from the controller.
* State query requests are defined by I\*ModelView based interfaces, where \* is replaced by model defined class names. State query requests are initiated from the view.
* Notifies the view of changes to the state of the model (e.g. Observer)

## View

* Renders the state of the application taken from the Model.
* Processes input (not necessarily from a user) as gestures to the Controller.
* Accepts state change notifications from the Model, using a push model (Observer) which may trigger the view to re-render the output.
* View classes implement an IView interface, allowing the controller to select different views.

## Controller

* Holds the behavioral logic of the application
* Selects/creates the views and maps user gestures into actions and state changes in the Model
* Implements the IGesture interface? (*not sure*)

## Common

* Holds common classes used by the other packages in the system
* Does not reference any of the other packages in the system (Model, View, Controller) avoiding a possible circular package dependency