The design pattern used is called MVC pattern which stand for Model-View-Controller. In this pattern, the code is divided into one or objects which defines each element of the pattern. To apply the MVC pattern, we use the following principles:

- The Model is used to implement the business logic and defines how the data is created and updated.

- The View defines the UI controls. Each view should serve some singular purpose for which it is being designed.

- The Controller is used to implement the view logic and is used to define how the information is displayed and how the user interacts with the UI components (what actions to take on user evnets such as click, drag, etc).

Since the UI, business logic and view logic are separated in this type of design It allows for more modularity in the program since the code is no loger tightly coupled to each other. The main advantage of this design is that the code is cleaner, easier to understand and the separation of logic helps developers to change any component without It affecting other functionality.

In our project, the MVC pattern is implemented in the following manner:

- The Model consists of GameManager, Config, GAmeObject, IMageLabel and TextLabel classes. These classes defines the variables, properties and behaviour that will be used throughout the program via the controller class.

- The View consists of the GameView class defines the UI controls such as VBox, ScrollPane, etc.

- The Controller consists of GamePanel class which has the objects for view and model classes and acts as an intermediary between the two. It also defines what the view elements behaviours will be such as drag detected and drag dropped and also implements the functionlaity for what action to take when an event is triggered and updated the model accordingly.

Based on the design, the view and model are separate classes with no way of interacting with each other and knowing each other states. The controller class is the one which declares the object for both view and model and acts as the intermediary between the two. This helps in separating the separate types of logic and model from each other. It is much easier to update any component without worrying about the others. For example, changing the behavior of user events user does not have to change the view but only the controller class.