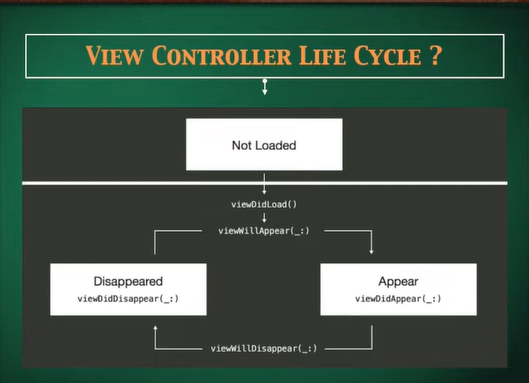
**Lifecycle and Methods of a View Controller**

In Swift, the lifecycle of a view controller refers to the sequence of events that occur from its creation to its removal from memory. Understanding the view controller lifecycle is crucial for managing the state of your app's user interface and performing necessary setup and cleanup tasks at the appropriate times. The lifecycle events allow you to respond to changes in the view controller's state and provide a way to coordinate actions with the system.

Here's an overview of the typical lifecycle events of a view controller in Swift:

* **Initialization:** When you create a view controller instance either programmatically or from a storyboard, its init method is called. You can perform any setup tasks in this method.
* **View Loading:** After initialization, if the view controller is loaded from a storyboard, its view is loaded from the storyboard file. The loadView method is called to load the view from the storyboard or to create it programmatically if loadView is overridden.
* **View Controller Configuration:** Once the view is loaded, the system calls the viewDidLoad method. This is where you perform additional setup tasks related to the view controller's view, such as configuring UI elements or loading data.
* **View Layout:** After viewDidLoad is called, the view controller's view is added to the view hierarchy, and its size and position are adjusted based on its container and constraints. This triggers the viewWillAppear and viewDidAppear methods, indicating that the view is about to appear on the screen and has appeared on the screen, respectively.
* **View Disappearance:** When the view controller's view is about to be removed from the screen, the viewWillDisappear and viewDidDisappear methods are called. These methods are often used to perform cleanup tasks or save the current state of the view controller.
* **Memory Management:** If the system determines that the view controller's view is no longer needed, it may remove the view from the view hierarchy to free up memory. At this point, the view controller's deinit method is called, allowing you to perform any final cleanup tasks before the view controller is deallocated.

Here's a visual representation of the view controller lifecycle:



Understanding these lifecycle events and their order helps you manage the state of your view controller and ensure that your app behaves correctly throughout its execution. You can override these methods in your view controller subclass to add custom behavior as needed at each stage of the lifecycle..