```swift
import Foundation
import UnityFramework

class Unity: UIResponder, UIApplicationDelegate {

    static let shared = Unity()

    private let dataBundleId: String = "com.unity3d.framework"
    private let frameworkPath: String = "/Frameworks/UnityFramework.framework"

    private var ufw : UnityFramework?
    private var hostMainWindow : UIWindow?

    private var isInitialized: Bool {
        ufw?.appController() != nil
    }

    func show() {
        if isInitialized {
            showWindow()
        } else {
            initWindow()
        }
    }

    func setHostMainWindow(_ hostMainWindow: UIWindow?) {
        self.hostMainWindow = hostMainWindow
    }

    private func initWindow() {
        if isInitialized {
            showWindow()
            return
        }

        guard let ufw = loadUnityFramework() else {
            print("ERROR: Was not able to load Unity")
            return unloadWindow()
        }

        self.ufw = ufw
        ufw.setDataBundleId(dataBundleId)
        ufw.register(self)
        ufw.runEmbedded(
            withArgc: CommandLine.argc,
            argv: CommandLine.unsafeArgv,
            appLaunchOpts: nil
        )
    }

    private func showWindow() {
        if isInitialized {
            ufw?.showUnityWindow()
        }
    }

    private func unloadWindow() {
        if isInitialized {
```

```
            ufw?.unloadApplication()
        }
    }

    private func loadUnityFramework() -> UnityFramework? {
        let bundlePath: String = Bundle.main.bundlePath + frameworkPath

        let bundle = Bundle(path: bundlePath)
        if bundle?.isLoaded == false {
            bundle?.load()
        }

        let ufw = bundle?.principalClass?.getInstance()
        if ufw?.appController() == nil {
            let machineHeader = UnsafeMutablePointer<MachHeader>.allocate(capacity: 1)
            machineHeader.pointee = _mh_execute_header

            ufw?.setExecuteHeader(machineHeader)
        }
        return ufw
    }
}

extension Unity: UnityFrameworkListener {

    func unityDidUnload(_ notification: Notification!) {
        ufw?.unregisterFrameworkListener(self)
        ufw = nil
        hostMainWindow?.makeKeyAndVisible()
    }
}
```