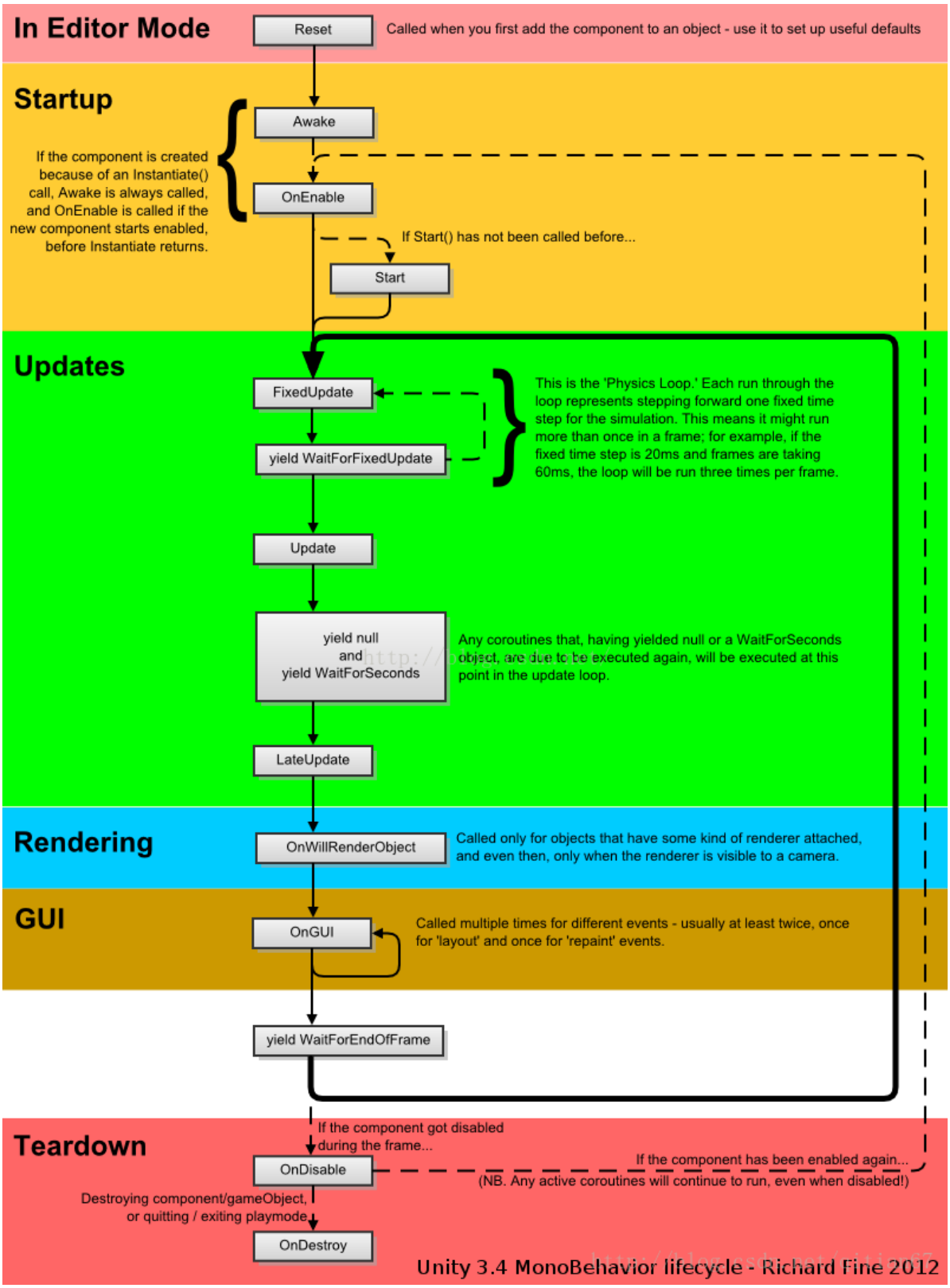
# Unity 生命周期




## Awake

在游戏开始前初始化变量或游戏状态，当**脚本实例被加载时被调用**，且在游戏对象被初始化后调用

在脚本的整个生命周期内它仅被调用一次

每个物体上的 `Awake` 以随机的顺序被调用

不能执行协同程序

`Awake` **只会被调用一次**，无论之后是否被 `OnEnable` 都只会调用一次

如果 gameObject 的初始状态为关闭，那么运行程序时 `Awake` 不会被调用

`Awake` 的执行与脚本实例的状态无关，而是与脚本实例所绑定的游戏对象的状态相关

如果游戏实例被关闭再激活，则 `Awake` 不会被调用

# OnEnable

如果 `this.enable` 为 `false`，则禁用了当前脚本，则程序会跳转到 `OnDisable`

# Start

**尽在第一次 Update 函数调用之前被调用**

`Start` 在整个脚本生命周期内只被调用一次

`Start` 只在脚本实例被启用时调用

如果脚本实例被禁用了，那么 `Start` 函数不会被执行，只有在脚本实例被启用时它才会执行

可以理解为 `Awake` 是在脚本加载时调用，而 `Start` 是在脚本执行时调用

由于 `Awake` 是在游戏对象初始化后调用，所以一般在 `Awake` 函数中获取游戏对象的组件或脚本实例

然后在 `Start` 函数中进行一些初始化的设置

# Update

Update 分为 Fixed Update, Update 和 Late Update

Fixed Update 将会按照固定时间调用，用于物理上的更新

Update 将会在每一帧进行调用

Late Update 将会在所有其他 Update 调用后被调用，一般用于摄像机跟随更新

# OnGUI

在渲染和处理GUI事件时被调用

`OnGUI` 也是每一帧执行一次

# OnDisable

当物体被销毁时调用 `OnDisable`，可用于任意清理代码

脚本被卸载时，`OnDisable` 将会被调用

## OnDestory

当 Monobehaviour 将被销毁时，这个函数被调用

## 自定义执行顺序

在Unity开发中，不同的脚本可能会互相调用，这样需要保证脚本按照顺序初始化

虽然脚本的执行顺序可以通过挂载的顺序修改，但是当挂载的脚本多了以后就很难记住具体的顺序

此时可以通过 `Execution Order` 来更改脚本执行的顺序，保证每个需要的脚本都按顺序调用

## 参考

Unity生命周期

Unity脚本生命周期与执行顺序

Unity中Awake与Start函数的调用情况总结