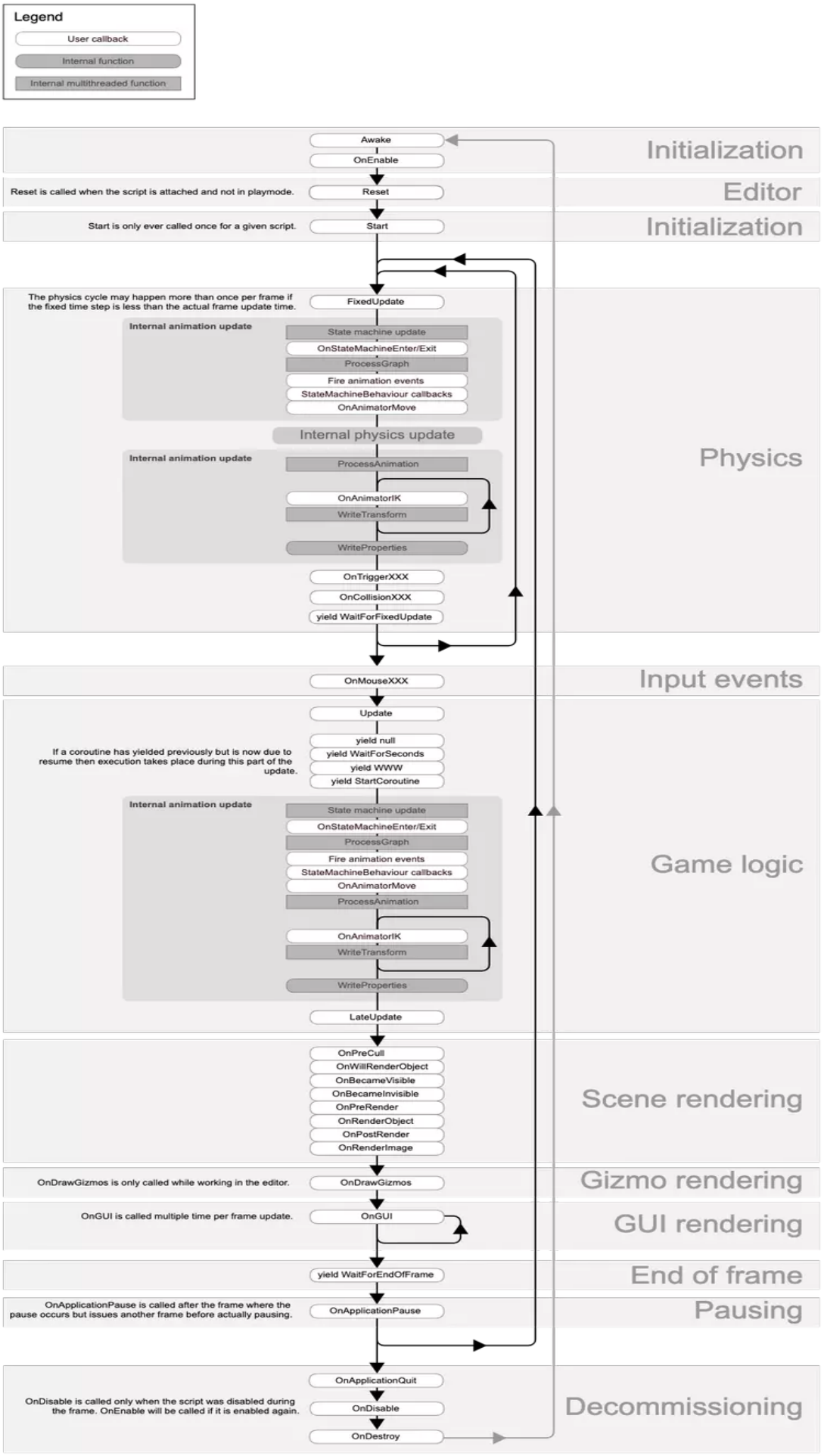

Legend
User callback
Internal function
Internal multithreaded function
Initialization
Awake
OnEnable
Editor
Reset is called when the script is attached and not in playmode.
Reset
Initialization
Start is only ever called once for a given script.
Start
Physics
The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time.
FixedUpdate
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
Internal physics update
Internal animation update
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
OnTriggerXXX
OnCollisionXXX
yield WaitForFixedUpdate
Input events
OnMouseXXX
Game logic
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
LateUpdate
Scene rendering
OnPreCull
OnWillRenderObject
OnBecameVisible
OnBecameInvisible
OnPreRender
OnRenderObject
OnPostRender
OnRenderImage
Gizmo rendering
OnDrawGizmos is only called while working in the editor.
OnDrawGizmos
GUI rendering
OnGUI is called multiple time per frame update.
OnGUI
End of frame
yield WaitForEndOfFrame
Pausing
OnApplicationPause is called after the frame where the pause occurs but issues another frame before actually pausing.
OnApplicationPause
Decommissioning
OnApplicationQuit
OnDisable is called only when the script was disabled during the frame. OnEnable will be called if it is enabled again.
OnDisable
OnDestroy

Mở đầu

- Trước tiên để tìm hiểu về các hàm được cung cấp sẵn, mình sẽ đưa ra sơ đồ tổng quát của *Unity3D Documentation* sau.
- Mặc định trong bài viết các gameObjects được active cùng một lúc nhé

I. Initialization

Như chúng ta thấy ở bảng trên, Awake, OnEnable và Start nằm cùng một phần gọi là Initialization nhưng công dụng của chúng khác nhau như thế nào?, và chúng được gọi khi nào trong vòng đời của MonoBehaviour?

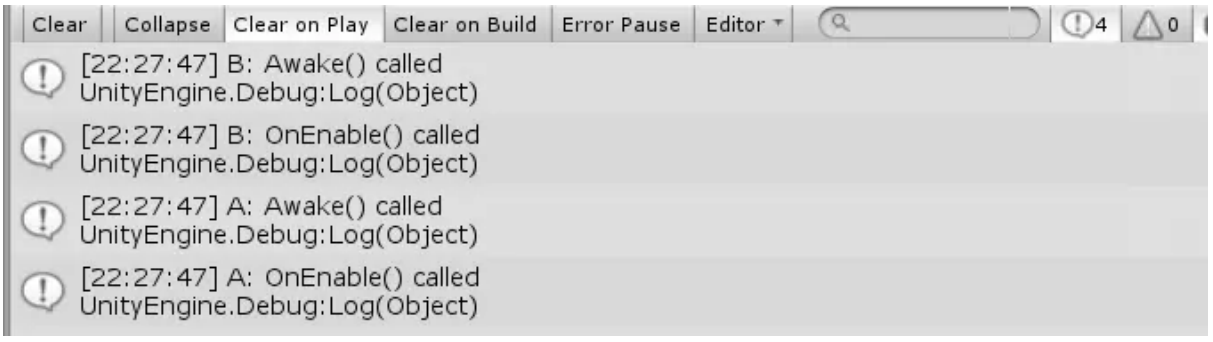
- Awake(): các hàm Awake đều được gọi trước TẤT CẢ các hàm Start của các *MonoBehaviour* có trong Scene (được enable cùng lúc) hay gọi ngay sau khi gameObject đó được sinh ra bằng hàm Instantiate().

Awake chỉ được gọi một lần duy nhất trong vòng đời của một *MonoBehaviour*

- OnEnable(): hàm này được gọi khi một gameObject được đổi trạng thái từ *deactive -> active* hoặc khi enable Component và được gọi lần đầu tiên ngay sau hàm Awake của nó.

"Vậy nếu có 2 gameObjects A và B thì thứ tự gọi có phải là Awake A -> Awake B -> OnEnable A -> OnEnable B?"

Không, vào lúc active lần đầu tiên, thứ tự gọi như sau:



Thứ tự gọi của Awake() và OnEnable() của 2 gameObjects bất kỳ

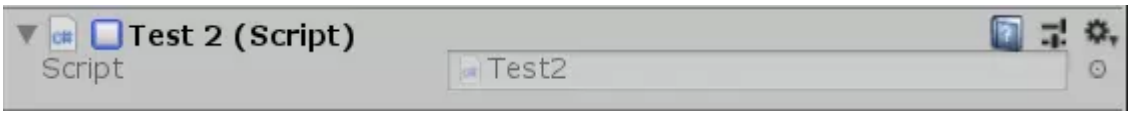
“Sao Awake của B gọi trước Awake của A vậy?”

Thứ tự gọi Awake trong Unity là random.

- Start(): hàm Start() được gọi sau OnEnable, trước khi các frames bắt đầu chạy hay trước các hàm Update. Cũng như Awake, nó chỉ được gọi một lần duy nhất.

“Hàm Start còn có gì khác so với Awake nữa không?”

Có một điểm đặc biệt của hàm Awake, nó sẽ được gọi kể cả khi script KHÔNG được enable (xem ảnh dưới), trong khi Start thì sẽ không được gọi. Tuy nhiên trong trường hợp GameObject de-active thì không có hàm nào được gọi.



Checkbox enable-disable script

Như vậy, các hàm thuộc loại Initialization thường được sử dụng để khởi tạo các giá trị có sẵn hoặc cache các components (GetComponent), trong đó chỉ có OnEnable() có thể được gọi nhiều lần

II. Game Logic

Ở phần này mình sẽ không đề cập về Internal Animation Update, ta chỉ quan tâm đến phần này khi sử dụng StateMachineBehaviour.
Đại loại nếu MonoBehaviour được gắn vào vào các GameObject thì StateMachineBehaviour sẽ được gắn vào các animation state và cũng có các hàm tương tự để xử lý

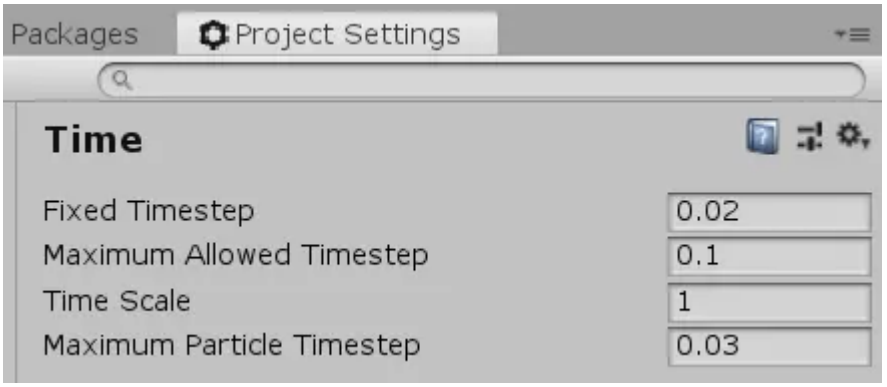
- Update(): chắc hẳn hàm này đã quá quen thuộc khi chúng ta mới bắt đầu sử dụng Unity engine, Update được gọi mỗi lần vào mỗi frame và được xem như hàm xử lý chính của vòng lặp game.
- LateUpdate(): LateUpdate() được gọi như Update, mỗi frame một lần, điểm khác biệt duy nhất là nó được gọi sau khi các hàm Update của tất cả các MonoBehaviour đã được thực thi xong.

Một chức năng kinh điển của hàm này là khi sử dụng camera follow theo player, sau khi vị trí cũng như rotation của player đã được tính toán hoàn tất trong Update, thì ta sẽ LateUpdate vị trí và góc xoay của camera theo player, đảm bảo được độ chính xác cao.

yield Null, WaitForSeconds, WWW, StartCoroutine,… được thực thi tiếp khi sử dụng ở phần Coroutine, có thể đọc bài Coroutine trong Unity là gì? để tham khảo thêm.

III. Physics

- FixedUpdate: Hàm này được gọi không phụ thuộc vào vòng lặp chính của game mà gọi theo vòng lặp của vật lý trong game, thường nó được gọi cố định vào mỗi 0.02s theo mặc định của project setting, con số này gọi là Fixed Timestep hay fixedDeltaTime.



Điều chỉnh Fixed Timestep trong Project Settings

Vì vậy FixedUpdate có thể được gọi nhiều hơn một lần hoặc không gọi trong 1 frame tùy theo FPS của game

- Internal Physics Update: Đây là vùng xử lý chính của Physics System trong unity được xử lý riêng mà chúng ta không can thiệp, với một số chức năng sẵn có như áp trọng lực, trigger các hàm va chạm, di chuyển các vật thể sử dụng vật lý (rigidbody, joints,…)

Vùng này được thực thi sau FixedUpdate, vì vậy các xử lý liên quan tới vật lý thì chúng ta nên đặt ở trong FixedUpdate thay vì Update để Physics System của Unity có thể cập nhật vị trí và xét va chạm kịp thời.

Mình đã nói qua vấn đề lỗi này ở bài Cơ bản về Rigidbody.

IV. Decommissioning

- OnApplicationQuit(): cái tên cũng nói lên được công dụng của hàm này rồi, nó sẽ được gọi trước khi thoát game (hoặc tắt Play mode trong Unity Editor)
- OnDisable(): hàm này tương ứng với OnEnable mà mình đã đề cập ở trên, có thể được gọi nhiều lần nếu GameObject active-deactive nhiều lần (hoặc enable-disable Component gắn vào GameObject)
- OnDestroy(): cuối cùng là OnDestroy, hàm này sẽ được gọi ở cuối frame, đồng nghĩa với việc khi xóa một GameObject trong scene (bằng script hoặc các sự kiện như thoát game, chuyển scene,…) hay sử dụng hàm Destroy(gameObject), nó sẽ không xóa ngay lập tức mà phải đến cuối frame đó nó mới được dọn dẹp.

Để dọn dẹp một GameObject ngay lập tức mà không cần delay tới cuối frame, có thể sử dụng DestroyImmediate, tuy nhiên Unity khuyến cáo là chỉ nên sử dụng khi debug và không nên sử dụng trong game.