FALMOUTH
UNIVERSITY

3: Software Architecture

GAM140: Individual Creative Computing Project

Reminder – This week!

- **Proposal Review**
  - **Describe** the game design that will form the basis for your interface
  - **Illustrate** basic research into electronic component and physical form factors for controllers
  - **Analyse** the design of the controller in detail;
  - **List** the key electronic components of your controller
  - and **List** the key user stories.

Learning outcomes

- **Understand** basic architecture of a video game
- **Understand** what is meant by software quality
- **Develop** your Game/Controller proposal

Introduction

- Every game has a similar architecture
- This is governed by the interactive nature of video games
- **But** also the architecture of hardware
  - The GPU has to render something to the screen
  - We have to update the game state before this

The Game Loop

- The most basic game loop does **three** things:
  1. Handle **Input**
  2. **Update** the state of the game
  3. **Render** the game to the screen
- It does these actions **once per frame** (30 or 60 times per second)

The Game Loop

```
bool running = true;

while (running)
{
        handleInput();
        update();
        render();
}
```

Handling input

- At this stage of the update you should grab the input from the user
  - **Input.GetButtonDown("Fire1")**
  - **Input.GetKeyDown(KeyCode.Space)**
  - **Input.mousePosition**
- If you are using **Input.GetButton*** or **Input.GetAxis**, these have to be configured in the **Input Manager**
- You can use the values from the input to update the state of the game
- You should also consider checking out:
  - **Rewired** - https://guavaman.com/projects/rewired/
  - **InControl** - http://www.gallantgames.com/pages/incontrol-introduction
  - **Unity New Input System** - https://blogs.unity3d.com/2019/10/14/introducing-the-new-input-system/

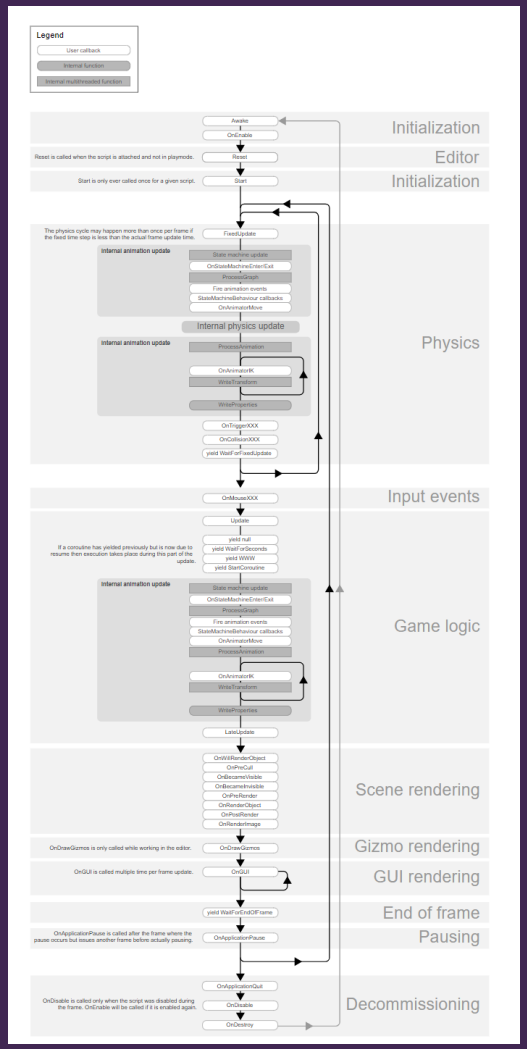
Updating the game state

- This is where your **game logic** is implemented
  - Physics, AI, Game play
- In Unity the update is split in two, a physics update (**FixedUpdate**) and a game update (**Update**)

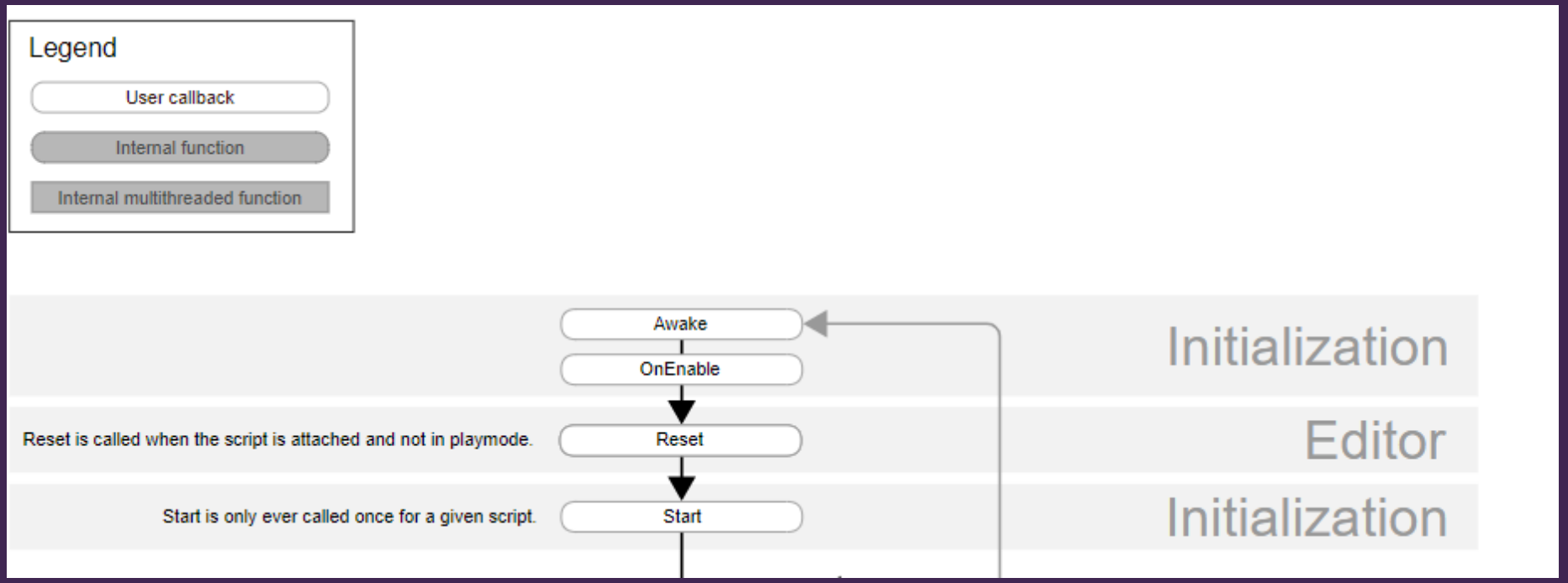
Rendering

- This is where the current state of the game is drawn
- Anything visual will be rendered to the backbuffer
- When drawing is complete, the backbuffer and the frontbuffer is swapped
- This will display the current state of the game to the player

Unity Engine Architecture

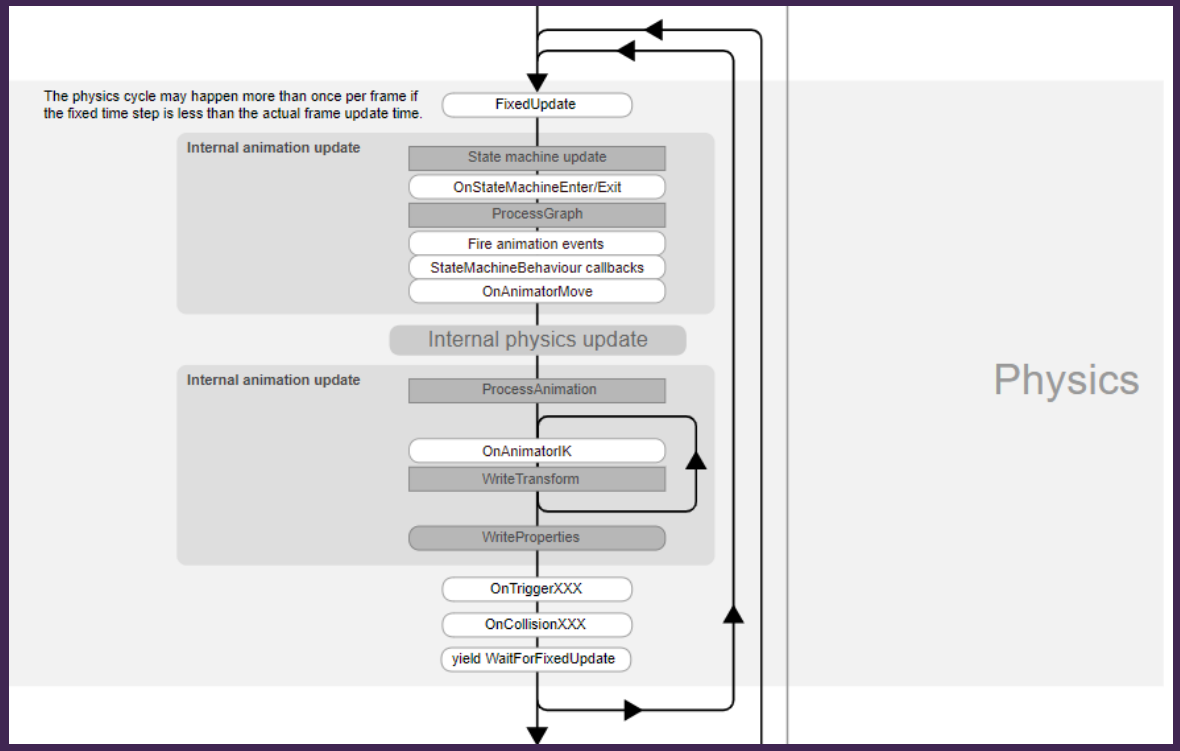


Unity Engine Architecture



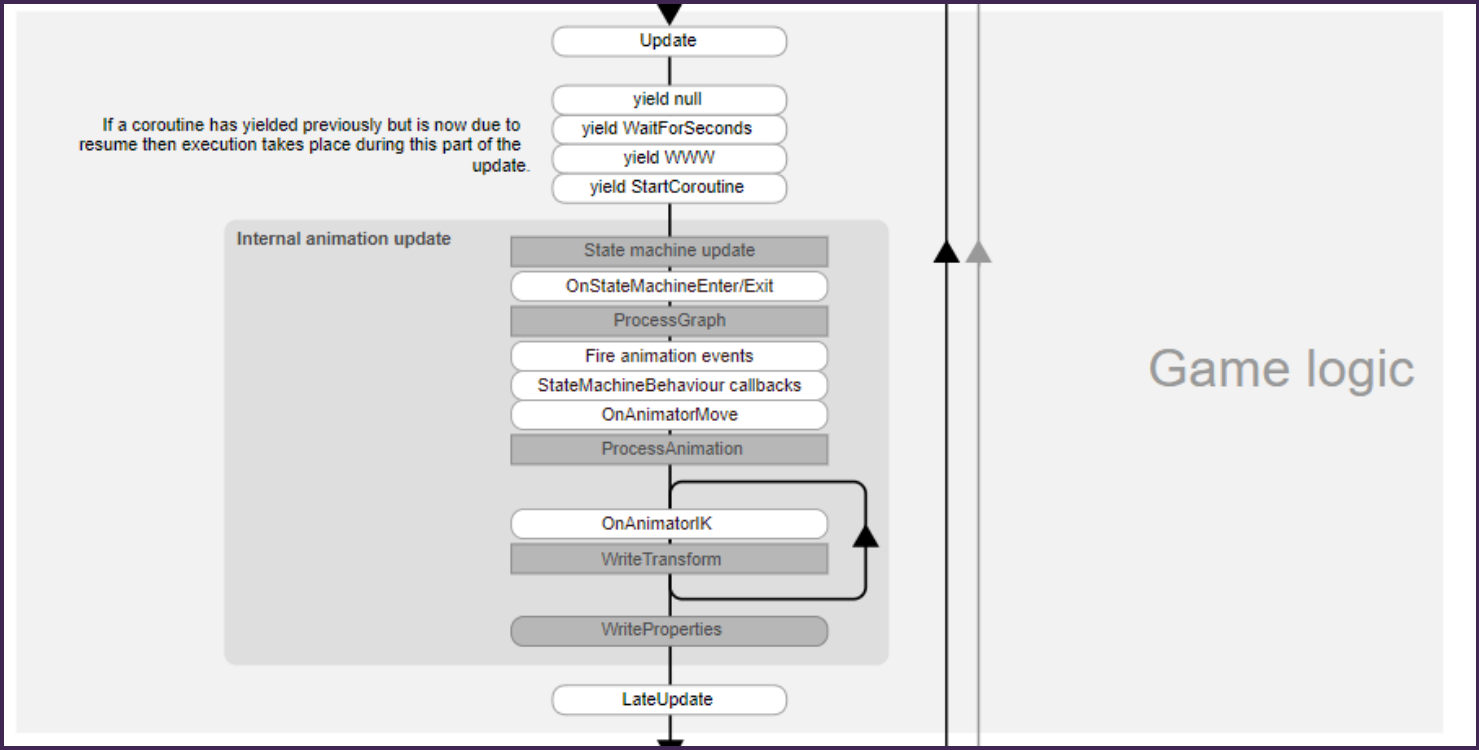
- On loading of a scene, the following gets called
  - **Awake** : Called before any **Start**
  - **OnEnable** : Called before any Game Object is enabled
  - **OnLevelWasLoaded** : Called when the level is loaded

Unity Engine Architecture



- You should perform all physics calculations in **FixedUpdate**
- You don’t need to multiply any by **deltaTime** in **FixedUpdate**

Unity Engine Architecture



- One thing to note is that **Update** gets called before all **Coroutines** and **Animation** Updates
- **LateUpdate** : This is called before any drawing occurs

Unity Engine Architecture - Notes

- Under the hood, Unity adds all GameObjects to a list and then iterates through
- Calling functions in the order mentioned above
- However, there is no fixed order on how **Update** will be called on each Game Object in the scene
- If you require some order, have a look at **Script Execution** order
- Or consider using **LateUpdate**