Làm quen với DOTS

Mục tiêu

Kết thúc hướng dẫn này bạn có thể:

- Hiểu các phần của Data-Oriented Technology Stack (DOTS) trong Unity.
- Xác định trường hợp có lợi khi sử dụng DOTS.

Tổng quan

Hiệu năng CPU và bộ nhớ là những thành phần quan trọng cần cân nhắc để đem tới trải nghiệm tốt khi phát triển ứng dụng. Data-Oriented Technology Stack là kiến trúc phần mềm cho phép lập trình viên tận dụng tốt nhất phần cứng của họ.

DOTS là tập hợp công nghệ đem tới một cách khác để làm việc với Unity. Nó là cách tiếp cận khác khi nghĩ về code, dữ liệu sử dụng data-oriented design DOD thay vì object-oriented design (OOP). DOTS cho phép bạn tận dụng CPU đa lõi để xử lý dữ liệu song song. Với DOTS, bạn có thể tạo ra ứng dụng với hiệu năng tốt hơn và sử dụng phần cứng tốt hơn.

Hướng dẫn này dành cho cả người biết và không biết về kỹ thuật, là người mới với DOTS. Cung cấp hiểu biết cơ bản về DOTS và giải đáp các câu hỏi thường gặp khi tìm hiểu về DOTS.

Các package và tính n?ng.

DOTS là tập hợp các gói và tính năng cho phép viết code hiệu suất cao:

- **C# Job System** mang tới cách đơn giản và an toàn để viết code đa luồng tận dụng CPU đa lõi ngày nay.
- **Burst Compiler** là compiler tối ưu hóa cho C# tạo ra code nhanh hơn Mono hoặc ngay cả ILCPP.
- **Unity.Mathematics** mang tới thư viện toán được tối ưu đặc biệt khi sử dụng với code sử dụng Burst-Compiler.
- **Unity.Collections** cung cấp các collection phổ biến như Lists và hashmaps. Không giống như C# collections, collections này không được quản lý nên có thể sử dụng trong code Burst-Compiled. Collections này cũng hỗ trợ kiểm tra an toàn giúp bạn chắc chắn rằng chúng được sử dụng đúng trong jobs.
- **Unity.Entities** cung cấp một dạng kiến trúc Entity Component System (ECS). Ngắn gọn thì entities là một thay thế nhẹ hơn, có hiệu quả hơn cho GameObject, và chúng được xử

lý bởi các đơn vị code gọi là systems.
- **Unity.Entities.Graphics** renders entities sử dụng URP (The Universal Rendering Pipeline) hoặc HPRP (The High-Definition Rendering Pipeline).

Một số package khác được xây dựng dựa trên Entities và lõi DOTS:

- **Unity.Physics** đem tới hệ thống vật lý. *(nguyên văn: The Unity.Physics package provides a deterministic rigid body dynamics system and spatial query system.)*
- **Unity.Netcode** (được biết đến là **Netcode for Entities**) cung cấp hệ thống multiplayer qua mạng với một máy chủ có thẩm quyền và một client có thể dự đoán.

Hiện tại, Unity chưa có giải pháp cho *animation*, *audio*, hay *UI* dựa trên Entities vậy nên các dự án dựa trên entities phải trở lại với GameObject hoặc các cách thay thế khác cho những tính năng này. Ví dụ, nến bạn tạo một game dựa trên entities với nhân vật được tạo anim thì bạn có thể xử lý nhân vật dưới dạng entities trong logic nhưng hiển thị nó như một GameObjects. Điều này đòi hỏi đồng bộ trạng thái giữa từng entity nhân vật và GameObject của nó, điều này (được Unity cho rằng) là có thể chấp nhận được với các dự án ở quy mô vừa và nhỏ.

Các khái niệm chính của DOTS.

Các video dưới đây giới thiệu nhanh về các khái niệm cơ bản trong DOTS:

- The C# Job system (11 minutes).
- ECS Entities and components (10 minutes).
- ECS Systems (7 minutes).

Có nên sử dụng DOTS?

Bất kỳ dự án nào bị nghẽn cổ chai CPU trong code đều nên được xem xét triển khai lại với slow code như Burst-Compiled jobs. Burst-Compiled code không chỉ nhanh hơn Mono-Compiled nhiều lần hay thậm chí tương đương IL2CPP-compiled, jobs còn cho phép bạn chia nhỏ công việc cho tất cả các lõi của CPU.

Tin tốt là Burst-Compiled jobs còn thể được tích hợp vào một dự án có sẵn. Ngay cả khi dự án đó không sử dụng DOTS. Bạn có thể sẽ cần copy dữ liệu vào và ra khỏi Unity.Collections nhưng mặt khác thêm Burst-Compiler jobs thường không yêu cầu tái cấu trúc code một cách nghiêm ngặt.

Điều này ít đúng hơn với package Entities. Mặc dù đôi khi có thể tích hợp có chọn lọc các entities để thực hiện các tính năng cụ thể, kiến trúc ECS áp đặt các cấu trúc code riêng của nó và do đó thường tạo nên nền tảng chung cho dự án của bạn.

Dưới đây là 4 lý do để sử dụng Entities cho dự án của bạn:

- Dự án của bạn có nhiều thành phần tĩnh, chẳng hạn như render môi trường rộng và chi tiết. Trong dự án mẫu Megacity thể hiện một môi trường phức tạp được tạo thành từ các entity.
- Dự án có nhiều thành phần động, có thể bao gồm các hành vi tính toán nặng, ví dụ những game RTS thường cần tính toán đường đi cho hàng trăm thậm chí hàng ngàn đơn vị.
- Bạn thích cấu trúc dữ liệu và code của ECS hơn. Cách này được cho là dễ hiểu và dễ bảo trì hơn OOP.
- Bạn đang tạo game multiplayer có tính cạnh tranh với những hành động nhanh như bắn súng, do đó bạn yêu cầu một máy chủ có thẩm quyền và khả năng dự đoán phía máy khách để tạo ra trải nghiệm tốt cho người chơi. (Những tính năng này được hỗ trợ bởi Netcode for Entities còn Netcode for GameObjects thì không).

Hướng dẫn của khoá học này.

Ba hướng dẫn của khoá học này bao gồm các hướng dẫn các sử dụng cơ bản của Entities và job system.

Hướng dẫn đầu tiên, “Bắt đầu với job system của Unity”, trình bày cách sử dụng C# job system, Burst, Unity.Collections và Unity.Mathematics để tạo một giải pháp hiệu năng cao cho một vấn đề liên quan đến tính toán nặng trên CPU.

Hướng dẫn thứ 2, “Học entities với HelloCube”, đi qua các ví dụ vô cùng cơ bản để tạo và sử dụng entities.

Hướng dẫn thứ 3, “Entities hành động: Tanks”, giới thiệu một mô phỏng rất đơn giản về việc di chuyển và bắn đạn của khẩu pháo.

Tài liệu bổ sung.

Để tìm hiểu nhiều hơn về DOTS, tham khảo website.

Revision #2
Created 2025-08-18 04:04:42 UTC by ThanhDV
Updated 2025-08-18 04:09:20 UTC by ThanhDV