Lý do USN được công khai dưới dạng mã nguồn mở và ý định thiết kế

Lời nói đầu

Ngày 26/10/2021, Haruki Yano đã release Unity Screen Navigator dưới dạng mã nguồn mở.

20211027174149.gif
Những vấn đề kỹ thuật đã được tác giả viết trong file README.md. Vậy nên trong bài viết này tác giả sẽ tập trung vào lý do mà USN được công khai dưới dạng mã nguồn mở và các vấn đề liên quan đến thiết kế. Ngoài ra, lưu ý rằng bài viết có thể chứa nhiều ý kiến cá nhân của tác giả.

Unity Screen Navigator (USN) là gì?

USN là một thư viện để thực hiện các chức năng **chuyển đổi màn hình**, **hoạt ảnh chuyển đổi màn hình**, **quản lý lịch sử chuyển đổi** và **quản lý vòng đời các màn hình** trong Unity.

Tính năng:

- Chuyển đổi màn hình dễ dàng và linh hoạt.
- Quản lý vòng đời màn hình và bộ nhớ.
- Có thể triển khai các hoạt ảnh phức tạp.
- Các tính năng khác như quản lý lịch sử, ngăn chặn click...

Tại sao USN được công khai dưới dạng mã nguồn mở?

Lý do mà tác giả tham gia các hoạt động OSS không chỉ là để nâng cao kỹ năng chuyên môn hay phát triển sự nghiệp mà hơn thế, Haruki Yano luôn băn khoăn về thực trạng các cá nhân và công ty đều đang tự mình phát triển những thứ phổ biến và giống hệt nhau.

Dĩ nhiên, những bí quyết và công nghệ hiếm có thể tạo ra lợi thế cạnh tranh thì không nên công khai, nhưng Haruki Yano nghĩ những thứ phổ biến và có tính ứng dụng chung thì nên được chia sẻ rộng rãi hơn. Làm như vậy các công ty và cá nhân có thể được hưởng lợi từ việc giảm bớt công sức đầu tư, nhưng lợi ích cũng chỉ dừng lại ở đó mà thôi.

Haruki Yano có một mong muốn rằng, thay vì phải lo lắng về những điều nhỏ nhặt như vậy thì mọi người nên gọi "lợi thế cạnh tranh" là khả năng tạo ra những tựa game thú vị hơn và có giá trị cao hơn.

Một lần nữa Haruki Yano tin rằng các hoạt động OSS mang lại nhiều lợi ích cho cả cá nhân lẫn doanh nghiệp.

Tại sao lại là USN?

Đơn giản là vì Haruki Yano muốn dùng nó.

Bản thân Haruki Yano cũng là người luôn muốn tận dụng những gì có sẵn, nên ông đã tìm kiếm với hy vọng rằng chắc chắn sẽ có một thư viện chuyển màn hình nào đó thật sự tốt. Tuy nhiên, kể cả các sản phẩm trả phí, số lượng thư viện liên quan đến chuyển màn hình được công bố rộng rãi lại rất ít.

Haruki Yano nghĩ rằng do các yêu cầu kỹ thuật quá đa dạng khiến cho việc tạo một thư viện chung có thể đáp ứng được tất cả trở nên khó khăn. Thêm vào đó, ông cảm giác rằng các game di động của châu Á thường có các kiểu chuyển màn hình phức tạp, nên như cầu có lẽ không đồng đều mà tập trung ở một số khu vực nhất định.

Cũng với lý do tương tự, Haruki Yano nghĩ rằng có thể mong đợi một engine phổ biến như Unity sẽ bổ sung thêm các tính năng như thế này.

Từ đó, Haruki Yano quyết định sẽ tự mình xây dựng USN. Điểm hay của việc tự phát triển là có thể tự do quyết định các yêu cầu kỹ thuật. Ông cho rằng nếu mình giới hạn các yêu cầu ở một mức độ nhất định thì trong phạm vi đó ông có thể tạo ra một thư viện phổ biến.

Hơn nữa, Haruki Yano nghĩ rằng một thư viện như USN cũng sẽ mang lại giá trị đủ lớn cho cộng đồng nên ông đã quyết định phát triển nó với mục tiêu biến nó thành mã nguồn mở.

Ý nghĩa của việc xác định và tài liệu hóa các quy tắc chuyển màn hình.

Việc đặt ra các điều kiện tiên quyết cũng có lợi ích là các thông số kỹ thuật của thư viện sẽ trở thành các quy tắc cho việc chuyển màn hình.

Ví dụ, hãy xem xét một dự án mà ở đó việc chuyển màn hình do UI designer quyết định và programmer là người triển khai. Nếu không có quy tắc nào được đặt ra programmer sẽ phải lường trước tất cả những thay đổi trong tương lai và thiết kế một hệ thống tổng quát hơn so với những yêu cầu đã nhận. Tuy nhiên, việc những thay đổi đó vượt ra ngoài dự tính vẫn thường xuyên xảy ra.

Nếu có các quy tắc ngay từ đầu thì cả, UI Designer và Programmer chỉ cần làm việc trong khuôn khổ đó. Hơn nữa, nếu các quy tắc này được áp dụng như một chuẩn chung cho nhiều dự án thì khi chuyển sang một dự án mới dev cũng không cần phải xác định hay tìm hiểu lại các quy tắc.

Ngoài ra, dù đã có quy tắc nhưng nếu chúng không được thể hiện một cách rõ ràng, dễ hiểu để dev nắm bắt thì quy tác cũng sẽ trở nên vô nghĩa.

Chỉ khi các quy tắc này được tài liệu hóa và truyền đạt một cách bài bản thì các cuộc thảo luận về việc phát triển ứng dụng dựa trên các quy tắc mới có thể diễn ra.

Vì vậy, Haruki Yano tin rằng việc xác định và tài liệu hóa các quy tắc sẽ mang lại lợi ích to lớn cho quá trình phát triển.

Thư viện chuyển màn hình và mức độ trừu tượng.

Khi tiến hành triển khai, Haruki Yano đã rất băn khoăn về việc nên xây dựng thư viện này với cấu trúc và mức độ trừu tượng như thế nào.

Nếu muốn khái quát hóa một cách triệt để thì một thư viện chuyển màn hình sẽ trở thành một thư viện "*chuyển cái gì đó bằng cách nào đó tới phần nào đó ở màn hình*".

Ở đây, thứ gì đó có thể là Prefab, Scene,... còn bằng cách nào đó không chỉ dừng lại ở animation mà có thể material, effect...

Ta hoàn toàn có thể triển khai chức năng "*chuyển cái gì đó bằng cách nào đó tới phần nào đó ở màn hình*" như một lớp thực thi ở tầng thấp rồi xây dựng các tính năng khác trên đó, nhưng Haruki Yano quyết định không làm vậy vì nó sẽ trở nên khó hiểu và hơn hết là bản thân Haruki Yano không cần đến nó.

Kết quả là Haruki Yano quyết định xây dựng một thư viện cho uGUI, chủ yếu hỗ trợ prefab và được thiết kế ở mức mà các scene có thể sử dụng bằng cách mở rộng thêm. Haruki Yano cũng quyết định sẽ không xét đến những hiệu ứng chuyển cảnh đặc thù như material animation hay post-effect.

Sau cùng, tác giả nghĩ rằng ông đã đạt được mức độ trừu tượng vừa phải, có thể đáp ứng được khoảng 80% các yêu cầu kỹ thuật thông thường. Tất nhiên con số 80% cũng chỉ là áng chừng.

Sự chú trọng vào việc tách biệt workflow.

Một điểm nữa mà Haruki Yano chú trọng là tách biệt workflow.

Haruki Yano không thích luồng làm việc mà programmer sử dụng các thư viện Tween để tạo các animation. Khi làm việc với UI designer hoặc animator chuyên nghiệp, Haruki Yano vừa kinh ngạc trước chất lượng và sự tỉ mỉ trong các tác phẩm của họ vừa cảm thấy rằng programmer không nên bước chân vào lĩnh vực đó với kiến thức và kỹ năng nửa vời.

Ngoài ra, workflow mà UI designer, animator đưa ra hướng dẫn cho programmer thực hiện cũng trở thành lãng phí và gây căng thẳng cho cả 2 bên. Để tạo ra một sản phẩm tốt, giao tiếp là điều quan trọng, những việc xây dựng một workflow giúp loại bỏ những trao đổi không cần thiết cũng quan trọng không kém.

Dĩ nhiên, không thể tránh khỏi trường hợp mà programmer cần phải làm animation, nhưng workflow vẫn cần tách biệt nhiều nhất có thể.

Với các lý do dó, Haruki Yano đã thiết kế thư viện này với các yêu cầu cho phép tách biệt rõ ràng workflow.

Revision #4
Created 2025-06-23 05:09:04 UTC by ThanhDV
Updated 2025-06-23 16:46:25 UTC by ThanhDV