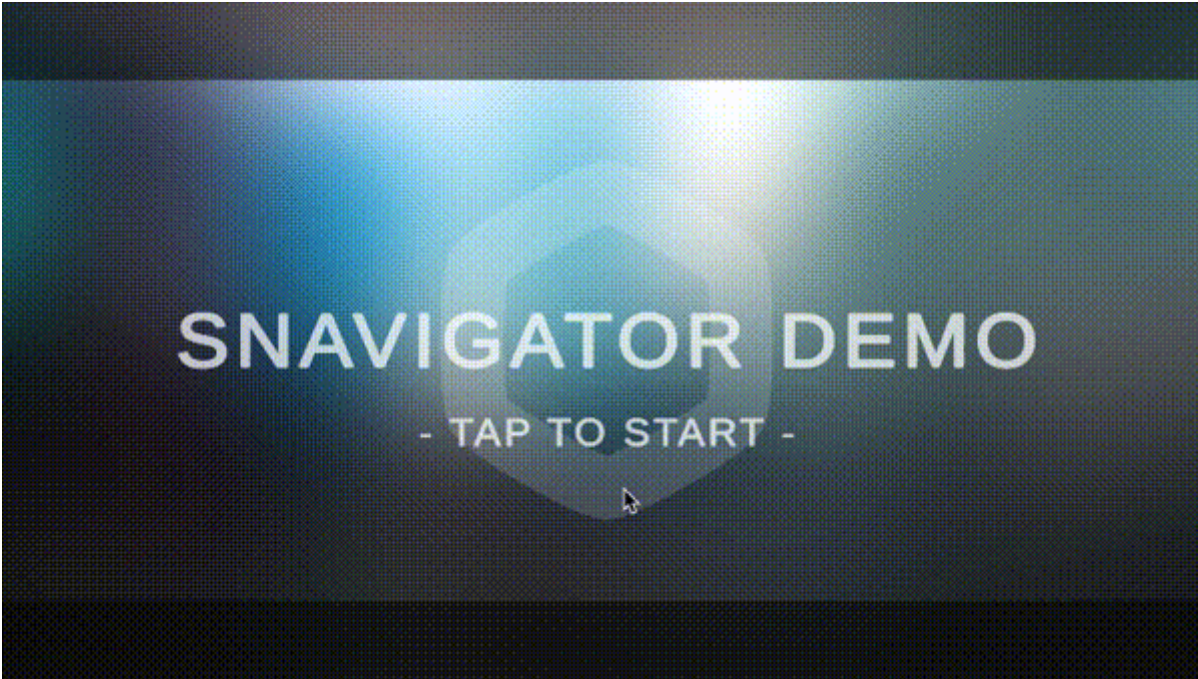


README.md

Unity Screen Navigator (trong bài này gọi tắt là USN) là thư viện cho screen transitions, transition animations, transition history stacking, và quản lý screen trong uGUI của Unity



Tổng quan

Tính năng

- Tạo pages, modals, tabs và chuyển động của chúng một cách dễ dàng và linh hoạt.
- Quản lý screen và bộ nhớ từ khi load cho đến khi destroy.
- Quy trình tách biệt với các animator cho các hoạt ảnh chuyển đổi màn hình phức tạp.
- Thư viện được phân tách tốt không có các chức năng phụ (ví dụ: thư viện GUI, máy trạng thái).
- Và các tính năng tiêu chuẩn như history stacking và ngăn chặn nhấp chuột trong quá trình chuyển đổi.

Demo

Bạn có làm theo những bước sau để chơi thử

1. Clone repo
2. Mở và chạy scene DemoEntryPoint

Cài đặt

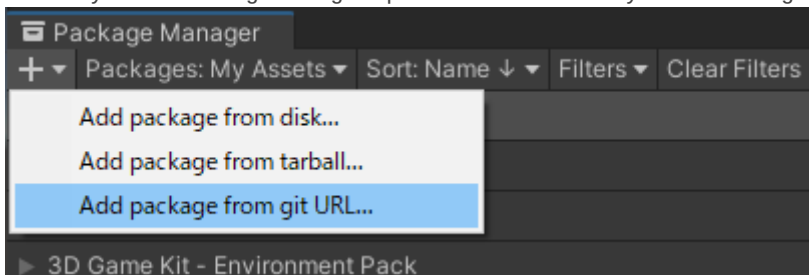
Yêu cầu

- Unity 2021.3 hoặc cao hơn
- uGUI (không hỗ trợ UIElements)

Cài đặt

Unity Package Manager

1. Mở Package Manager (Window > Package Manager).
2. Nhấn nút (+) > Add package from git URL.
3. Nhập `https://github.com/Haruma-K/UnityScreenNavigator.git?path=/Assets/UnityScreenNavigator` và nhấn Add.



Thêm file Manifest

Mở Packages/manifest.json và thêm dòng sau:

```
{
  "dependencies": {
    "com.harumak.unityscreennavigator": "https://github.com/Haruma-K/UnityScreenNavigator.git?path=/Assets/UnityScreenNavigator"
  }
}
```

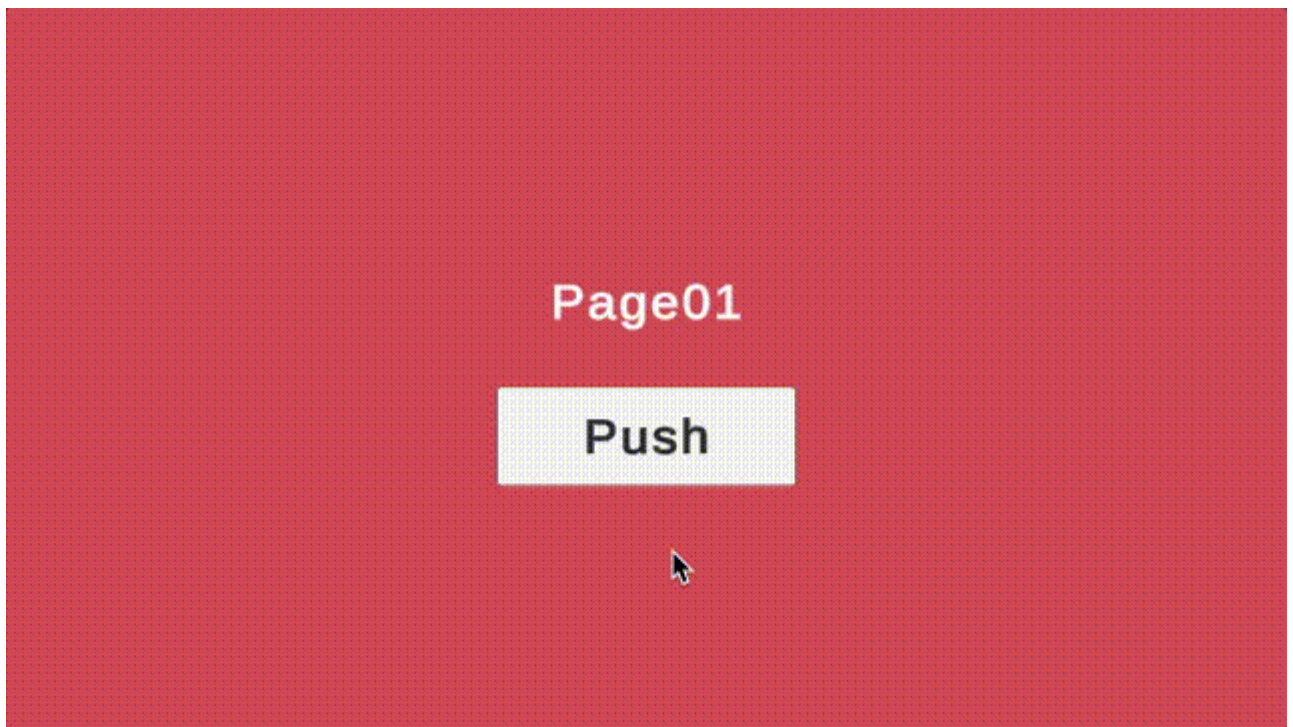
Nếu muốn cài phiên bản khác thì thêm số phiên bản vào cuối đường link (VD: #1.0.0)
`https://github.com/Haruma-K/UnityScreenNavigator.git?path=/Assets/UnityScreenNavigator#1.0.0`

Chuyển màn hình cơ bản

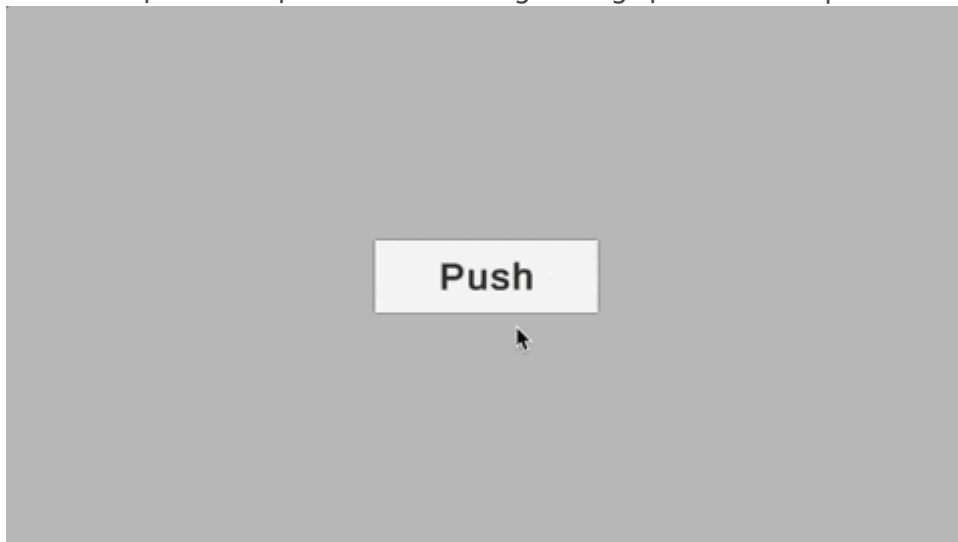
Khái niệm về màn hình (screens) và chuyển tiếp (transitions)

USN chia màn hình (screens) thành 3 loại: Page, Modal và Sheet.

- Page là một màn hình được chuyển đổi theo trình tự. (Nhìn giống các screen)
 - Khi chuyển từ Page A → Page B. Page A sẽ được stack trong lịch sử. Và khi từ Page B → Page A. Page A sẽ được hiển thị lại với trạng thái không đổi.



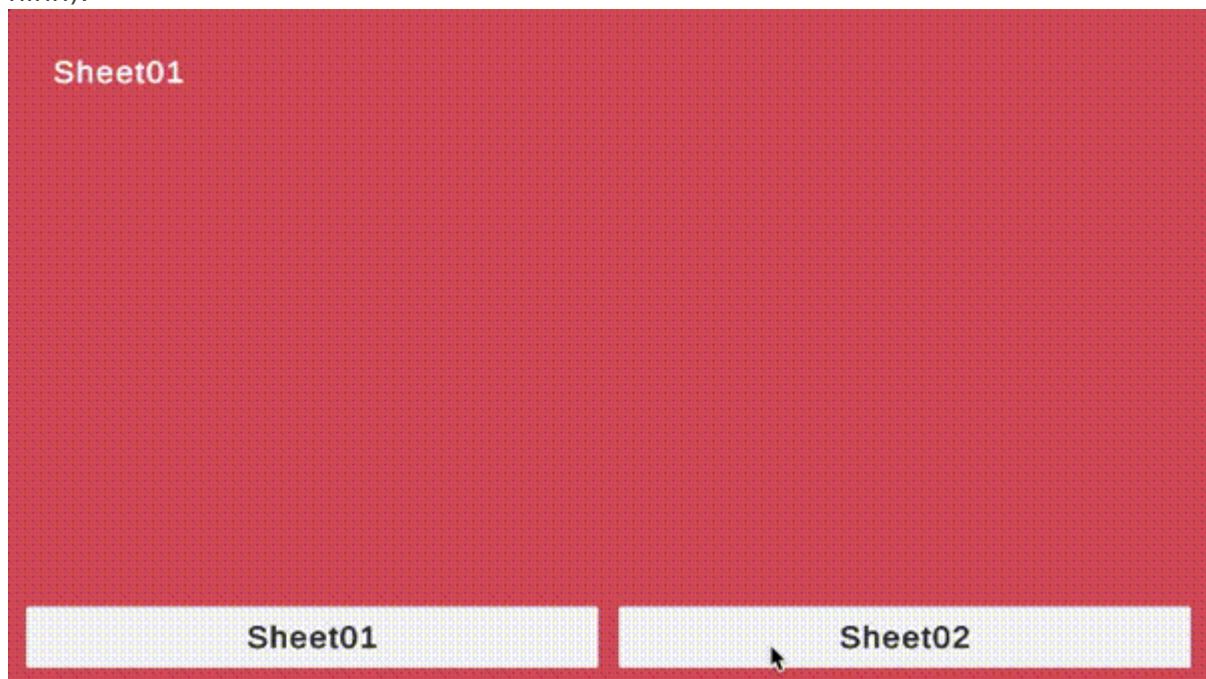
- Modal là một màn hình được stack trong một cửa window. (Nhìn giống popup)
 - Khi nó được hiển thị, tất cả các tương tác ngoại trừ modal phía đều bị chặn.



- Sheet được sử dụng giống như các tab.
 - Lịch sử không quản lý sheet và chỉ có một sheet được hiển thị cùng lúc.



- Các screen có thể được lồng vào nhau.
 - Và khu vực của mỗi màn hình có thể được chỉ định tự do (không bắt buộc full màn hình).



Tạo page

Để tạo page. Đầu tiên, tạo **Page Container** bằng cách gắn component PageContainer vào một GameObject bên trong Canvas. Pages sẽ hiển thị fit với container nên hãy điều chỉnh kích thước cho phù hợp.

Tiếp theo, tạo **Page** bằng cách tạo gameObject mới (tương ứng với một page) bên trong PageContainer và thêm component Page. Đặt GameObject này trong thư mục Resource với tên bất kỳ.

Và gọi `PageContainer.Push()` với đường dẫn Resources để hiển thị page.

Ví dụ cho push page được đặt tại `Assets/Resources/ExamplePage.prefab`.

```
PageContainer pageContainer;

// Push the page named "ExamplePage".
var handle = pageContainer.Push("ExamplePage", true);

// Wait for the transition to finish.
yield return handle;
//await handle.Task; // You can also use await.
//handle.OnTerminate += () => { }; // You can also use callback.
```

Gọi `PageContainer.Pop()` để deactivate page và hiển thị page trước đó.

```
PageContainer pageContainer;

// Pop the active page.
var handle = pageContainer.Pop(true);

// Wait for the transition to finish.
yield return handle;
```

Nếu muốn bỏ qua một trang nhất định khi gọi `Pop()`, có thể tắt *stacking history* bằng cách sử dụng tham số tùy chỉnh.

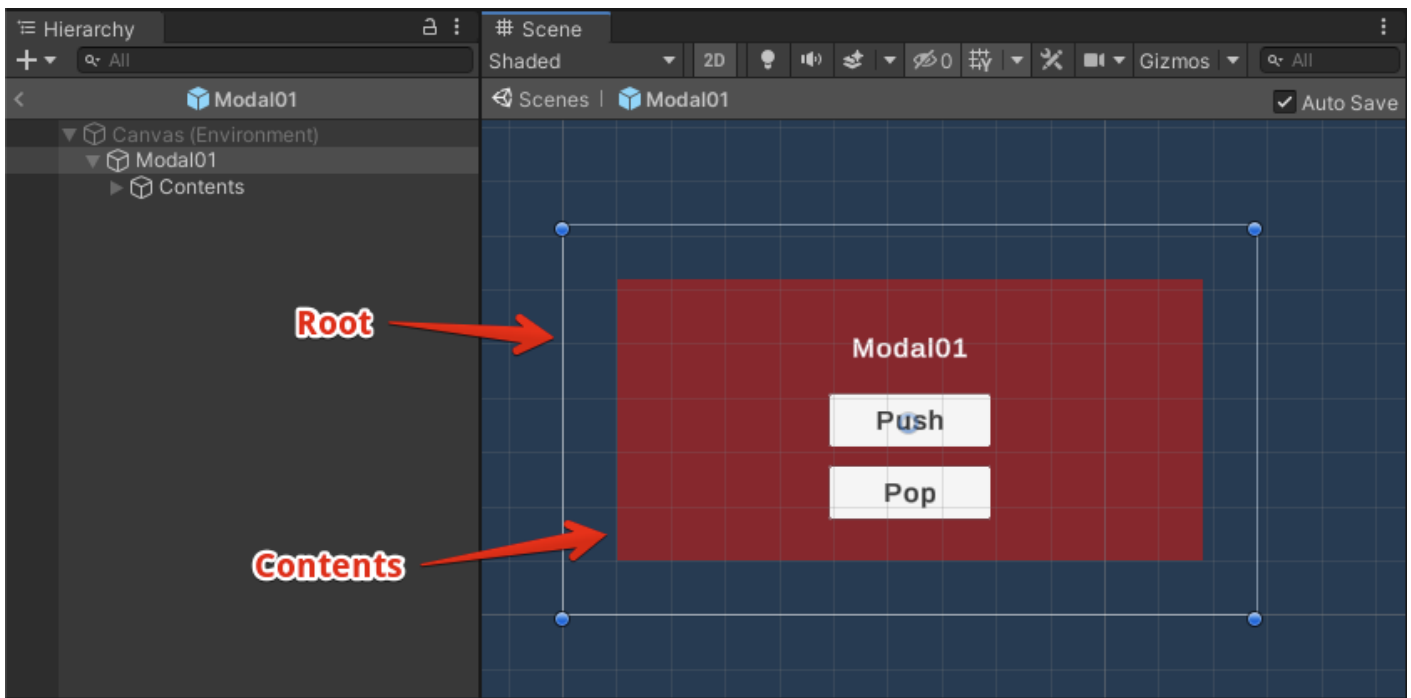
Tạo modal

Tương tự như tạo page, đầu tiên tạo một **Modal Container** bằng cách thêm component `ModalContainer` vào một gameObject bên trong Canvas.

Nhìn chung, các modal được thiết kế để che toàn bộ màn hình và chặn các click bằng phong nền của nó. Vì thế kích thước `RectTransform` của GameObject về cơ bản phải được đặt **khớp với kích thước màn hình**.

Tiếp theo, tạo Modal bằng cách thêm component `Modal` vào GameObject gốc của modal view. GameObject gốc này sẽ được điều chỉnh để fit với kích thước của Modal Container

Nếu bạn muốn tạo modal có margins, hãy tạo một GameObject con với kích thước nhỏ hơn.



Đặt Modal vào folder Resources với tên bất kỳ.

Và gọi `ModalContainer.Push()` với tham số là đường dẫn modal trong folder Resources để hiển thị modal.

Ví dụ cho hiển thị modal đặt tại `Assets/Resources/ExampleModal.prefab`

```
ModalContainer modalContainer;

// Push the modal named "ExampleModal".
var handle = modalContainer.Push("ExampleModal", true);

// Wait for the transition to finish.
yield return handle;
//await handle.Task; // You can also use await.
//handle.OnTerminate += () => { }; // You can also use callback.

Dùng ModalContainer.Pop() ?? deactivate modal và show modal tr??c ?ó.

ModalContainer modalContainer;

// Pop the active modal.
var handle = modalContainer.Pop(true);

// Wait for the transition to finish.
yield return handle;
```

Chú ý. Bạn có thể thay đổi phong nền của modal nếu muốn.

T?o Sheet

Tương tự như trên, đầu tiên thêm component `SheetContainer` vào game object bên trong Canvas. Hãy căn chỉnh kích thước của nó vì sheet sẽ được hiển thị fit với Canvas.

Tiếp theo, thêm component Sheet vào game object gốc của sheet view. Đặt game object này vào folder Resources với tên bất kỳ.

Gọi `SheetContainer.Register()` với đường dẫn trong folder Resources để tạo sheet. Sau khi được tạo, bạn có thể thay đổi active sheet bằng cách gọi `SheetContainer.Show()`. Khi đó, nếu có một sheet đang hoạt động thì nó sẽ bị deactivate.

Ví dụ, hiển thị sheet đặt tại `Assets/Resources/ExampleSheet.prefab`.

```
SheetContainer sheetContainer;

// Instantiate the sheet named "ExampleSheet"
var registerHandle = sheetContainer.Register("ExampleSheet");
yield return registerHandle;

// Show the sheet named "ExampleSheet"
var showHandle = sheetContainer.Show("ExampleSheet", false);
yield return showHandle;
```

Lưu ý rằng khi nhiều sheet có cùng key được khởi tạo bằng phương thức `Register()` thì các sheet không được đảm bảo key. Trong trường hợp này, hãy sử dụng ID của sheet thay vì key.

```
SheetContainer sheetContainer;

// Instantiate the sheet named "ExampleSheet" and get the sheet id.
var sheetId = 0;
var registerHandle = sheetContainer.Register("ExampleSheet", x =>
{
    sheetId = x.sheetId;
});
yield return registerHandle;

// Show the sheet with sheetId.
var showHandle = sheetContainer.Show(sheetId, false);
yield return showHandle;
```

Sử dụng `Hide()` để ẩn một sheet thay vì chuyển sheet khác.

```
SheetContainer sheetContainer;

// Hide the active sheet.
var handle = sheetContainer.Hide(true);

// Wait for the transition to finish.
yield return handle;
```

Làm thế nào để chuyển tiếp giữa các màn hình?

Mỗi phương thức (method) chuyển tiếp đều trả về `AsyncProcessHandle`. Sử dụng đối tượng này bạn có thể đợi các chuyển tiếp xử lý xong.

Bạn có thể dùng *coroutines*, phương thức *bất đồng bộ*, và *callback*

```
// Use coroutines
yield return pageContainer.Push("ExamplePage", true);
```

```
// Use async/await
await pageContainer.Push("ExamplePage", true).Task;

// Use callback
pageContainer.Push("ExamplePage", true).OnTerminate += () => { };
```

Lấy một container bằng static methods

Mỗi container (`PageContainer`/`ModalContainer`/`SheetContainer`) đều có các phương thức tĩnh để có thể lấy được.

Sử dụng `Container.Of()` bạn có thể lấy được các container gắn trên `Transform`/`RectTransform` gần nhất.

```
var pageContainer = PageContainer.Of(transform);
var modalContainer = ModalContainer.Of(transform);
var sheetContainer = SheetContainer.Of(transform);
```

Bạn cũng có thể dùng thuộc tính `Name` trong **Inspector** của container để lấy container bằng tên của nó. Trong trường hợp này, sử dụng `Container.Find()`.

```
var pageContainer = PageContainer.Find("SomePageContainer");
var modalContainer = ModalContainer.Find("SomeModalContainer");
var sheetContainer = SheetContainer.Find("SomeSheetContainer");
```

Animation chuyển màn hình

Cài đặt animation chung

Theo mặc định, có một animation tiêu chuẩn cho mỗi loại màn hình.

Bạn có thể tạo một class kế thừa `TransitionAnimationObject` để custom lại animation. Lớp này có các thuộc tính và phương thức để khai báo hành vi của animation.

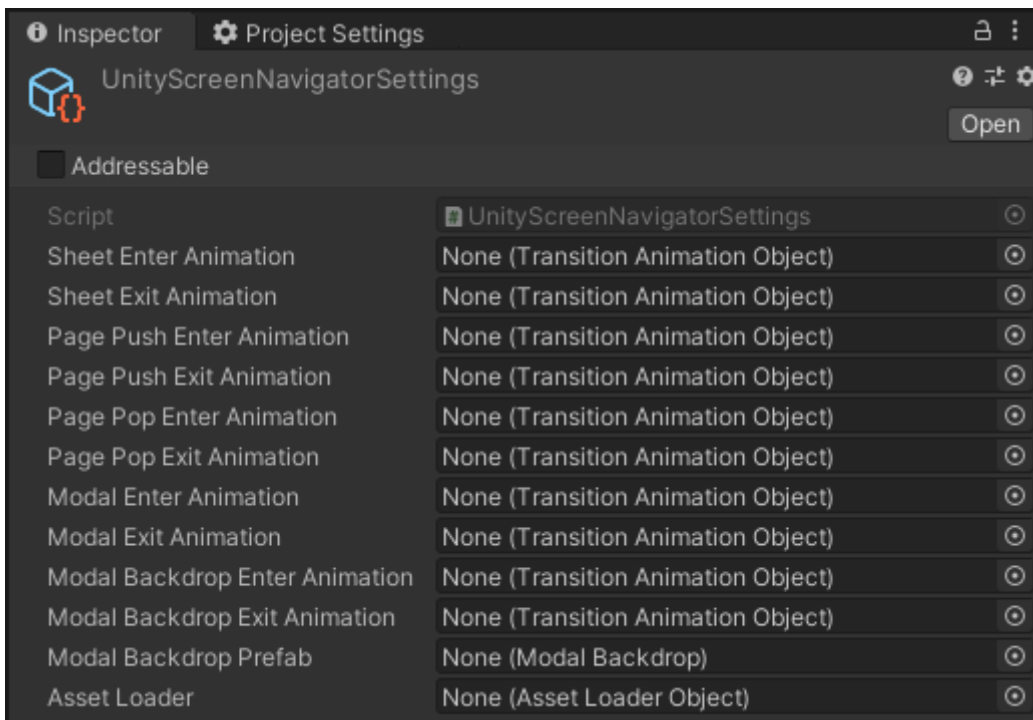
```
// Duration (sec).
public abstract float Duration { get; }

// Initialize.
public abstract void Setup();

// Define the state at this time.
public abstract void SetTime(float time);
```

Tham khảo `SimpleTransitionAnimationObject` để biết cách triển khai chi tiết.

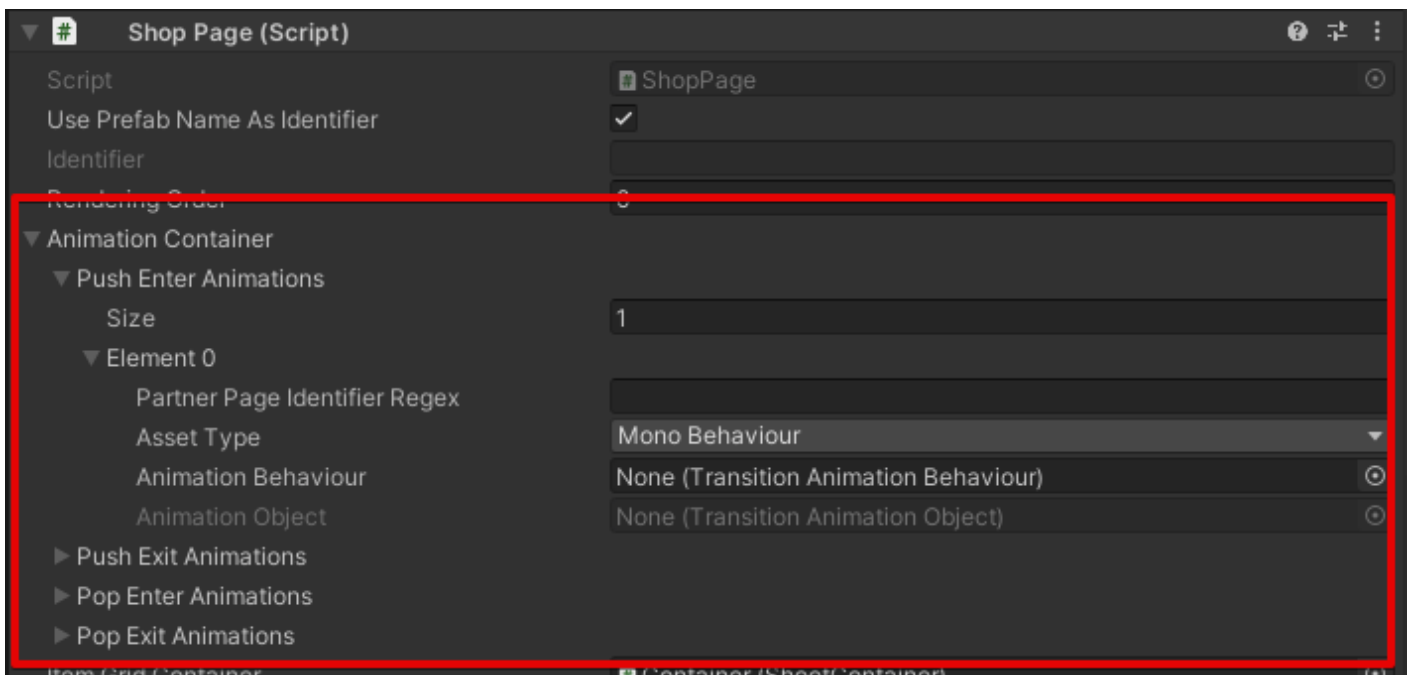
Sau đó, khởi tạo `ScriptableObject` này và gắn nó vào `UnityScreenNavigatorSettings`. Nếu chưa có `UnityScreenNavigatorSettings` bạn có thể tạo `Assets -> Create -> Screen Navigator Settings`.



Cài animation cho m?i màn hình

Bạn cũng có thể cài đặt animation khác nhau cho mỗi màn hình.

Mỗi Page, Modal, và Sheet đều có thuộc tính `AnimationContainer`. Bạn có thể gắn animation vào đó.



Bạn có thể thay đổi animation của màn hình này bằng cách chọn `AssetType` là `ScriptableObject` và gắn `TransitionAnimationObject` đã tạo ở bước trên cho `AnimationObject`.

Bạn cũng có thể sử dụng `MonoBehaviour` thay cho `ScriptableObject`. Trong trường hợp này bạn tạo một lớp kế thừa `TransitionAnimationBehaviour`.

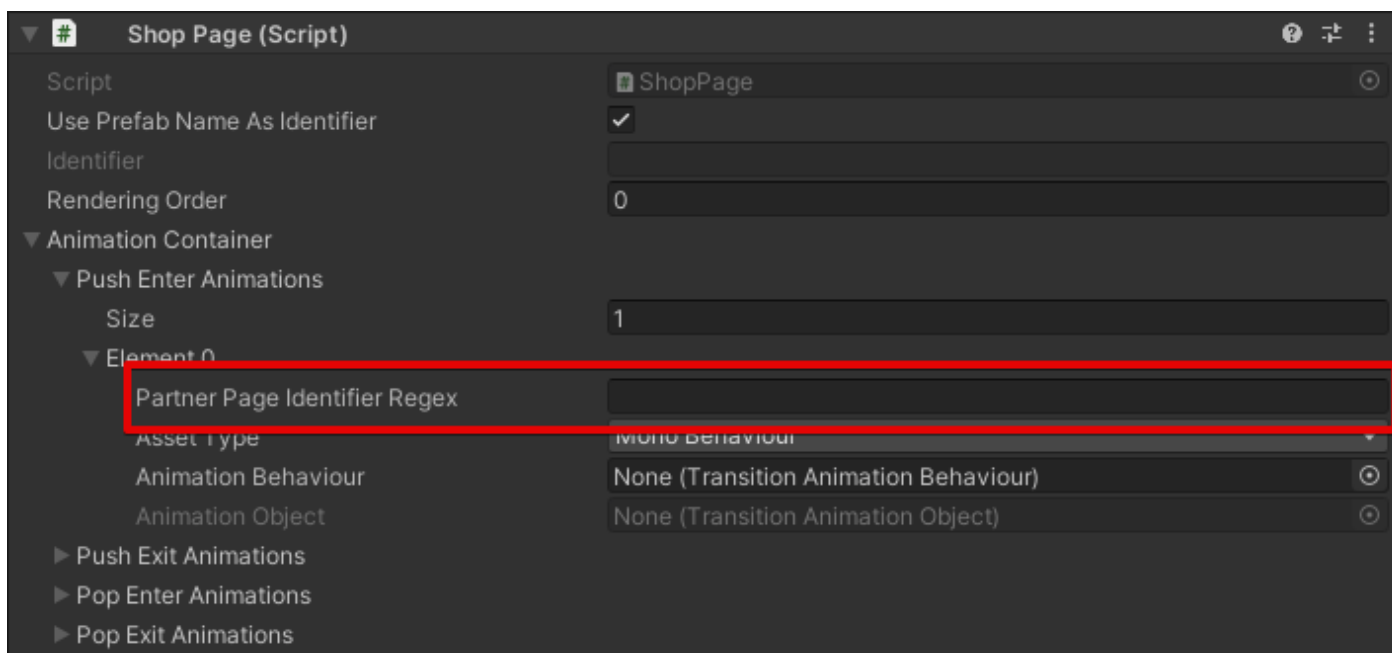
Xem `SimpleTransitionAnimationBehaviour` để biết cách triển khai chi tiết.

Sau đó có thể setup tương tự như trên. Chọn `AssetType` là `MonoBehaviour`, gán `TransitionAnimationBehaviour` vừa tạo ở trên vào `AnimationBehaviour`.

Thay thế hiệu ứng chuyển cảnh dựa trên màn hình tiếp theo.

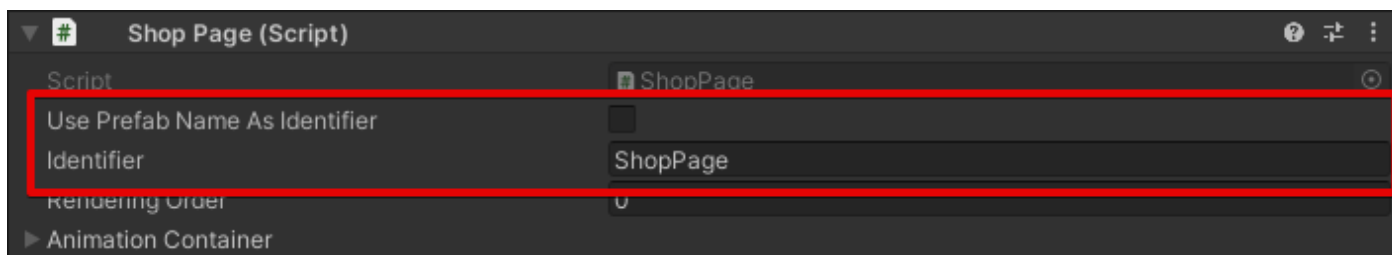
Ví dụ, khi vào màn hình A và thoát màn hình B, màn hình B được gọi là “màn hình đối tác” của màn hình A.

Nếu bạn nhập tên của màn hình đối tác cho thuộc tính bên dưới thì animation sẽ chỉ apply khi tên này trùng khớp với tên của màn hình đối tác.



Mặc định, tên của prefab được sử dụng như tên màn hình.

Nếu bạn muốn đặt tên cho nó. Hãy bỏ tick `UsePrefabNameAsIdentifier` và nhập tên vào ô `Identifier`.

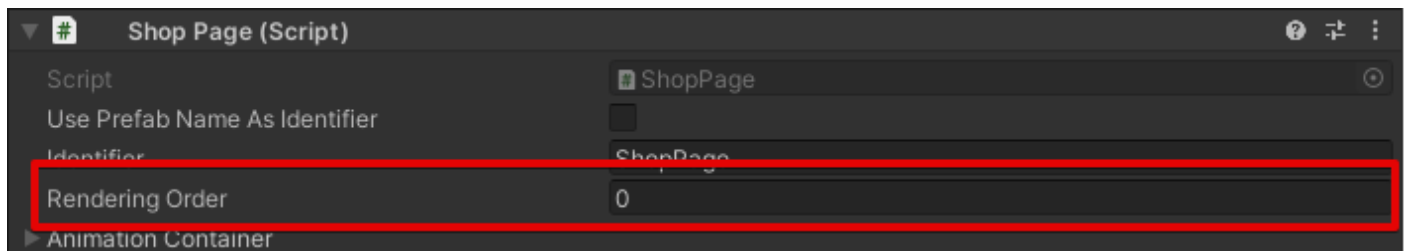


Ngoài ra, các biểu thức chính quy (regular expressions) có thể được sử dụng cho `PartnerPageIdentifierRegex`. Và nếu nhiều animation được đặt thì chúng sẽ được thực hiện từ trên xuống.

Hiệu ứng chuyển cảnh thế nào và thế nào?

Trong animation chuyển tiếp của một màn hình với đối tác của nó, thứ tự vẽ có thể quan trọng. Ví dụ một animation che mất màn hình đối tác.

Nếu bạn muốn control thứ tự này hãy sử dụng thuộc tính `RenderingOrder`.



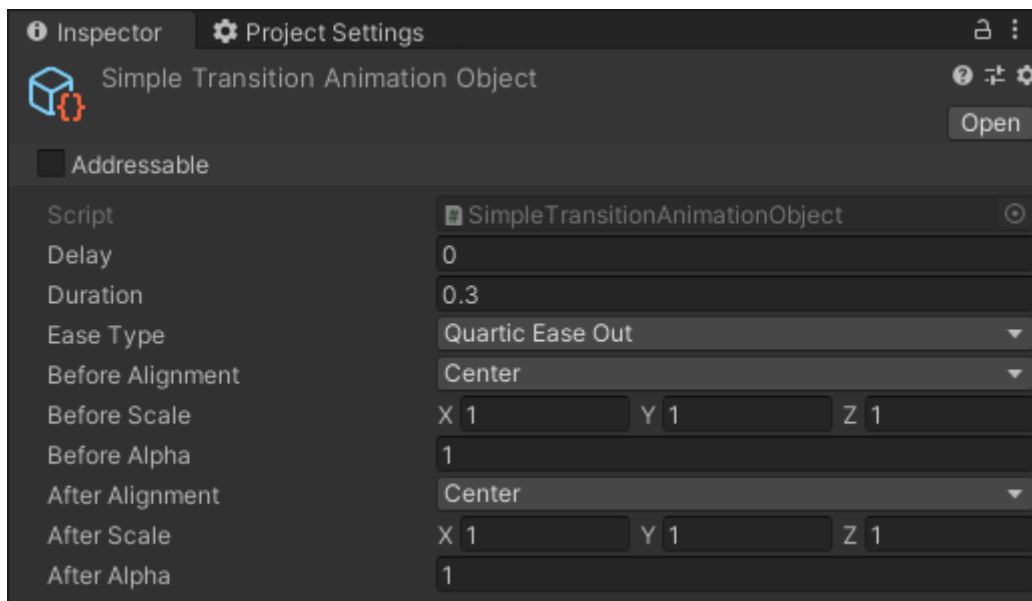
Trong quá trình chuyển màn hình, màn hình được vẽ theo thứ tự giảm dần của giá trị này.

Model không có thuộc tính `RenderingOrder` vì modal mới nhất luôn hiển thị bên trên.

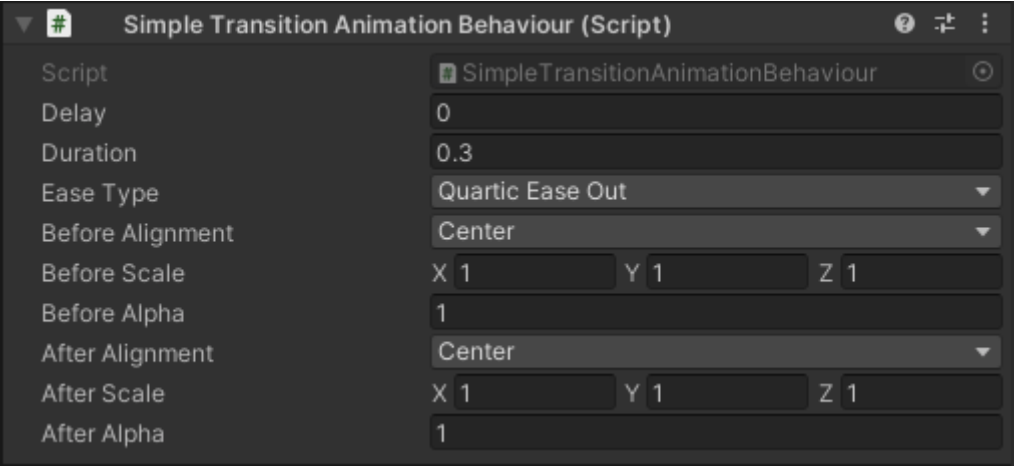
Tạo animation chuyển tiếp ??n gì?

Bạn có thể dùng `SimpleTransitionAnimationObject` như một triển khai đơn giản.

Bạn có thể tạo nó từ `Assets > Create > Screen Navigator > Simple Transition Animation`. Sau đó một `ScriptableObject` được tạo tương tự như hình bên dưới. Bạn có thể tùy chỉnh các thuộc tính trong cửa sổ Inspector.



Bạn cũng có thể dùng `SimpleTransitionAnimationBehaviour` như một triển khai `MonoBehaviour`. Nó được sử dụng bằng cách gán trực tiếp vào `GameObject`.



Giải thích các thuộc tính

Property Name	Description
Delay	Delay trước khi chạy animtion (giây).
Duration	Thời gian animation (giây).
Ease Type	Loại ease
Before Alignment	Vị trí tương đối so với container trước khi bắt đầu.
Before Scale	Scale trước khi bắt đầu.
Before Alpha	Độ trong suốt trước khi bắt đầu.
After Alignment	Vị trí tương đối so với container sau khi kết thúc.
After Scale	Scale khi kết thúc.
After Alpha	Độ trong suốt khi kết thúc.

Triển khai animation tổng tác với màn hình đối tác

Bạn cũng có thể tạo animation tham chiếu đến trạng thái của màn hình đối tác. Trong ví dụ dưới đây hình ảnh của modal trước đó được phóng to trong trong khi chuyển tiếp liền mạch sang modal tiếp theo.

```
137315378-974395a8-f910-41a9-8e07-2964efded848.qif
Để triển khai nó, đầu tiên tạo một lớp kế thừa từ TransitionAnimationObject hoặc
TransitionAnimationBehaviour.
Sau đó, tham chiếu tới thuộc tính PartnerRectTransform để lấy màn hình đối tác.
Nếu màn hình đối tác không tồn tại PartnerRectTransform sẽ trả về null.
```

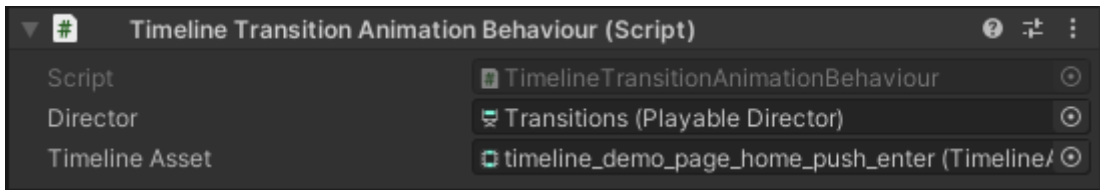
Xem CharacterImageModalTransitionAnimation để biết cách triển khai chi tiết.

Tạo animation với Timeline.

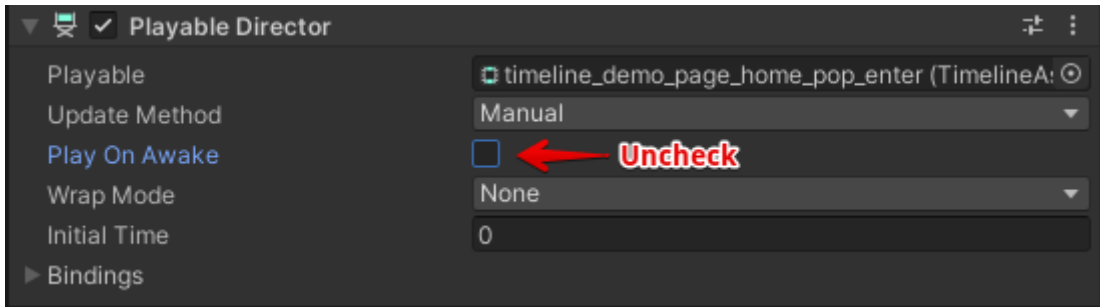
Bạn có thể sử dụng timeline để tạo animation. Chúng tôi khuyên bạn nên sử dụng timeline cho những animation phức tạp.

137634258-135b454e-04b5-49e8-a87a-bfb6ede03f49.gif

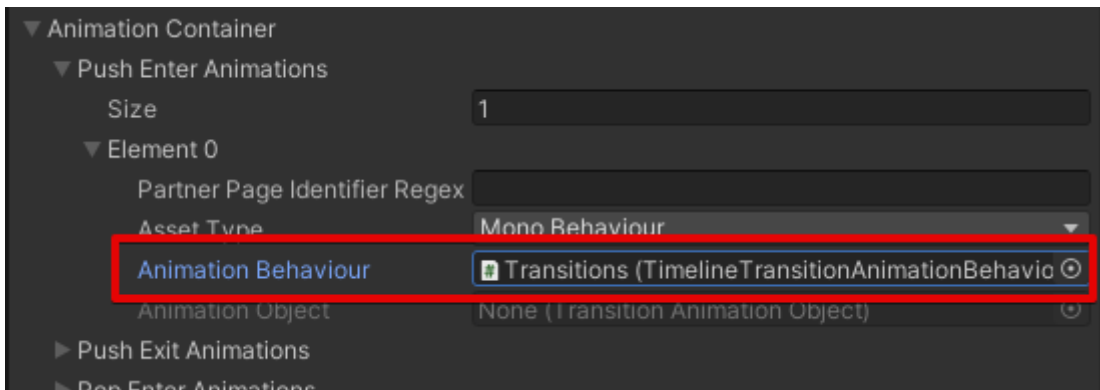
Để triển khai, đầu tiên thêm `TimelineTransitionAnimationBehaviour` vào `GameObject`. Sau đó gắn `PlayableDirector` và `TimelineAsset` cho các thuộc tính.



Chú ý bỏ chọn `PlayOnAwake` của `PlayableDirector`



Cuối cùng, gắn `TimelineTransitionAnimationBehaviour` vào `AnimationContainer`.



Ngoài ra, chúng tôi đề xuất bạn sử dụng `UnityUIPlayables` để tạo uGUI animation với Timeline.

113313016-cf9afe80-9345-11eb-9aa9-422c53b5a3f8.gif

Các sự kiện vòng đời (Lifecycle Events)

S? ki?n vòng ??i c?a page

```
using System.Collections;
using UnityEngine.SceneManagement;

public class SomePage : Page
{
    // Called just after this page is loaded.
    public override IEnumerator Initialize() { yield break; }
```

```

// Called just before this page is released.
public override IEnumerator Cleanup() { yield break; }
// Called just before this page is displayed by the Push transition.
public override IEnumerator WillPushEnter() { yield break; }
// Called just after this page is displayed by the Push transition.
public override void DidPushEnter() { }
// Called just before this page is hidden by the Push transition.
public override IEnumerator WillPushExit() { yield break; }
// Called just after this page is hidden by the Push transition.
public override void DidPushExit() { }
// Called just before this page is displayed by the Pop transition.
public override IEnumerator WillPopEnter() { yield break; }
// Called just after this page is displayed by the Pop transition.
public override void DidPopEnter() { }
// Called just before this page is hidden by the Pop transition.
public override IEnumerator WillPopExit() { yield break; }
// Called just after this page is hidden by the Pop transition.
public override void DidPopExit() { }
}

```

Bạn cũng có thể đăng ký thêm sự kiện bằng `Page.AddLifecycleEvents()` như sau.

```

// IPageLifecycleEvent is the interface that has lifecycle events described above.
// You can specify the execution priority with the second argument.
// Less than 0: executed before Page lifecycle event.
// Greater than 0: executed after Page lifecycle event.
IPageLifecycleEvent lifecycleEventImpl;
Page page;
page.AddLifecycleEvent(lifecycleEventImpl, -1);

// It is also possible to register only some lifecycle events as follows.
IEnumerator OnWillPushEnter()
{
    // Some code.
    yield break;
}
page.AddLifecycleEvent(onWillPushEnter: OnWillPushEnter);

```

Và bạn cũng có thể móc nối các sự kiện từ container bằng cách chuyển một object triển khai interface `IPageContainerCallbackReceiver` tới `PageContainer.AddCallbackReceiver()`.

```

public interface IPageContainerCallbackReceiver
{
    // Called just before the Push transition is executed.
    void BeforePush(Page enterPage, Page exitPage);
    // Called just after the Push transition is executed.
    void AfterPush(Page enterPage, Page exitPage);
    // Called just before the Pop transition is executed.
    void BeforePop(Page enterPage, Page exitPage);
    // Called just after the Pop transition is executed.
    void AfterPop(Page enterPage, Page exitPage);
}

```

Chú ý nếu bạn triển khai `IPageContainerCallbackReceiver` với `MonoBehaviour` và gán nó vào `GameObject` của container, nó sẽ đăng ký với `PageContainer` mà không cần gọi `PageContainer.AddCallbackReceiver()`.

Các sự kiện vòng đời của Modal

Bằng cách ghi đè các phương thức dưới đây trong class mà kế thừa lớp Modal. Bạn có thể viết lại các xử lý liên quan đến vòng đời của modal.

```
using System.Collections;
using UnityEngine.SceneManagement.Runtime.Core.Modal;

public class SomeModal : Modal
{
    // Called just after this modal is loaded.
    public override IEnumerator Initialize() { yield break; }
    // Called just before this modal is released.
    public override IEnumerator Cleanup() { yield break; }
    // Called just before this modal is displayed by the Push transition.
    public override IEnumerator WillPushEnter() { yield break; }
    // Called just after this modal is displayed by the Push transition.
    public override void DidPushEnter() { }
    // Called just before this modal is hidden by the Push transition.
    public override IEnumerator WillPushExit() { yield break; }
    // Called just after this modal is hidden by the Push transition.
    public override void DidPushExit() { }
    // Called just before this modal is displayed by the Pop transition.
    public override IEnumerator WillPopEnter() { yield break; }
    // Called just after this modal is displayed by the Pop transition.
    public override void DidPopEnter() { }
    // Called just before this modal is hidden by the Pop transition.
    public override IEnumerator WillPopExit() { yield break; }
    // Called just after this modal is hidden by the Pop transition.
    public override void DidPopExit() { }
}
```

Bạn cũng có thể đăng ký thêm các sự kiện với `Modal.AddLifecycleEvents()`

```
// IModalLifecycleEvent is the interface that has lifecycle events described above.
// You can specify the execution priority with the second argument.
// Less than 0: executed before Modal lifecycle event.
// Greater than 0: executed after Modal lifecycle event.
IModalLifecycleEvent lifecycleEventImpl;
Modal modal;
Modal.AddLifecycleEvent(lifecycleEventImpl, -1);

// It is also possible to register only some lifecycle events as follows.
IEnumerator OnWillPushEnter()
{
    // Some code.
    yield break;
}
modal.AddLifecycleEvent(onWillPushEnter: OnWillPushEnter);
```

Và bạn cũng có thể móc nối các sự kiện từ container bằng cách chuyển một object triển khai interface `IModalContainerCallbackReceiver` tới `ModalContainer.AddCallbackReceiver()`.

```
public interface IModalContainerCallbackReceiver
{
    // Called just before the Push transition is executed.
    void BeforePush(Modal enterModal, Modal exitModal);
    // Called just after the Push transition is executed.
    void AfterPush(Modal enterModal, Modal exitModal);
    // Called just before the Pop transition is executed.
    void BeforePop(Modal enterModal, Modal exitModal);
    // Called just after the Pop transition is executed.
    void AfterPop(Modal enterModal, Modal exitModal);
}
```

```
}
```

Chú ý rằng nếu bạn triển khai `IModalContainerCallbackReceiver` cho một `MonoBehaviour` và gán nó vào `GameObject` của `Container`, nó sẽ được đăng ký với `ModalContainer` mà không cần gọi `ModalContainer.AddCallbackReceiver()`.

Các sự kiện vòng đời của Sheet

Tương tự như trên bạn có thể ghi đè các phương thức sau của lớp `Sheet` để viết lại các xử lý liên quan đến vòng đời của `Sheet`.

```
using System.Collections;
using UnityEngine.SceneManagement.Runtime.Core.Sheet;

public class SomeSheet : Sheet
{
    // Called just after this sheet is loaded.
    public override IEnumerator Initialize() { yield break; }
    // Called just before this sheet is released.
    public override IEnumerator Cleanup() { yield break; }
    // Called just before this sheet is displayed.
    public override IEnumerator WillEnter() { yield break; }
    // Called just after this sheet is displayed.
    public override void DidEnter() { }
    // Called just before this sheet is hidden.
    public override IEnumerator WillExit() { yield break; }
    // Called just after this sheet is hidden.
    public override void DidExit() { }
}
```

Bạn cũng có thể đăng ký thêm sự kiện với `Sheet.AddLifecycleEvents()`.

```
// ISheetLifecycleEvent is the interface that has lifecycle events described above.
// You can specify the execution priority with the second argument.
// Less than 0: executed before Sheet lifecycle event.
// Greater than 0: executed after Sheet lifecycle event.
ISheetLifecycleEvent lifecycleEventImpl;
Sheet sheet;
Sheet.AddLifecycleEvent(lifecycleEventImpl, -1);

// It is also possible to register only some lifecycle events as follows.
IEnumerator OnWillEnter()
{
    // Some code.
    yield break;
}
sheet.AddLifecycleEvent(onWillEnter: OnWillEnter);
```

Và bạn cũng có thể móc nối các sự kiện từ container bằng cách chuyển một object triển khai interface `ISheetContainerCallbackReceiver` tới `SheetContainer.AddCallbackReceiver()`.

```
public interface ISheetContainerCallbackReceiver
{
    // Called just before the Show transition is executed.
    void BeforeShow(Sheet enterSheet, Sheet exitSheet);
    // Called just after the Show transition is executed.
    void AfterShow(Sheet enterSheet, Sheet exitSheet);
    // Called just before the Hide transition is executed.
}
```

```

void BeforeHide(Sheet exitSheet);
// Called just after the Hide transition is executed.
void AfterHide(Sheet exitSheet);
}

```

Và nếu bạn triển khai `ISheetContainerCallbackReceiver` với `MonoBehaviour` và gán nó với `GameObject` của container thì nó cũng sẽ được đăng ký với `SheetContainer` đó mà không cần gọi `SheetContainer.AddCallbackReceiver()`.

S? d?ng ph??ng th?c async thay cho coroutines

Bạn có thể sử dụng phương thức bất đồng bộ thay cho coroutines để định nghĩa các sự kiện vòng đời như bên dưới.

```

using System.Threading.Tasks;
using UnityScreenNavigator.Runtime.Core.Page;

public class SomePage : Page
{
    // Using asynchronous methods to define lifecycle events
    public override async Task Initialize()
    {
        await Task.Delay(100);
    }
}

```

Để sử dụng phương thức bất đồng bộ, hãy thêm Scripting Define Symbols như sau.

- Player Settings > Other Settings > Scripting Define Symbols
- Thêm `USN_USE_ASYNC_METHODS`

Chú ý rằng **Scripting Define Symbols** cần được set cho tất cả các nền tảng.

Load Screen Resource

Thay ??i ph??ng th?c load màn hình

Như đã nói ở trên, mặc định thì các màn hình sẽ được đặt trong folder Resources như một Prefab.

Nếu bạn muốn thay đổi phương thức tải. Đầu tiên hãy tạo Scriptable Object kế thừa `AssetLoaderObject`. `AssetLoaderObject` là một triển khai của `IAssetLoader` và có phương thức sau.

```

// Load the resource indicated by the key.
public abstract AssetLoadHandle<T> Load<T>(string key) where T : Object;

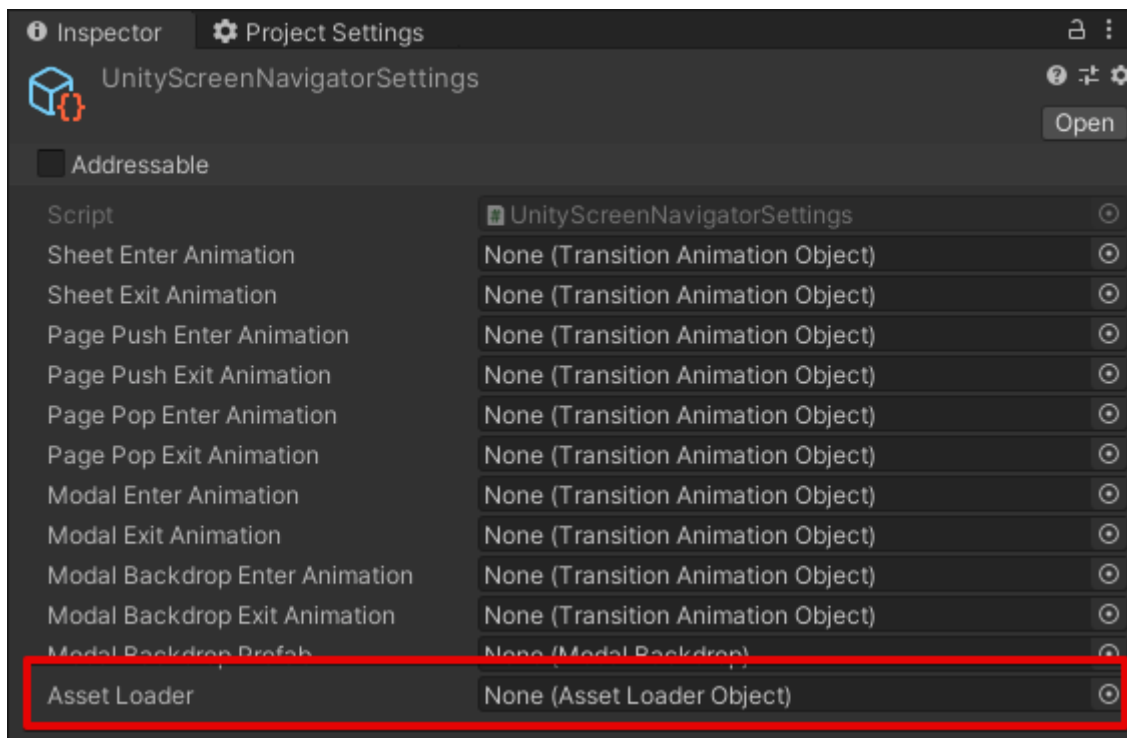
// Asynchronously load the resource indicated by the key.
public abstract AssetLoadHandle<T>

// Release the resource indicated by the handle.
public abstract void Release(AssetLoadHandle handle);

```

Xem thêm `ResourcesAssetLoader` để biết chi tiết cách triển khai.

Sau khi tạo, gán nó vào thuộc tính `AssetLoader` của `UnityScreenNavigatorSettings`.



Bạn có thể tạo `UnityScreenNavigatorSettings` từ `Assets > Create > Screen Navigator Settings`.

Bạn cũng có thể cài `IAssetLoader` cho mỗi container bằng cách set thuộc tính `AssetLoader` cho mỗi `Container`.

S? d?ng Addressable Asset System

Một triển khai của `IAssetLoader` cho Addressable Asset System được cung cấp mặc định. Nếu bạn muốn load màn hình với address. Thực hiện các bước sau:

- Chọn `Assets > Create > Resource Loader > Addressable Asset Loader`
- Gán `ScriptableObject` vừa được tạo vào thuộc tính `AssetLoader` của `UnityScreenNavigatorSettings`.

T?i ??ng b? (Load synchronously)

Bạn có thể truyền giá trị `false` cho tham số `loadAsync` của mỗi phương thức chuyển tiếp của container để tải đồng bộ các màn hình.

Ví dụ, `PageContainer.Push()` có thể được viết như sau:

```
PageContainer container;  
  
// Load synchronously.  
var handle = container.Push("FooPage", true, loadAsync: false);  
  
// Wait for the end of the transition animation.  
yield return handle;
```

Ngoài ra bạn cũng có thể khởi tạo nó trong cùng frame với lệnh gọi phương thức chuyển đổi bằng cách sử dụng callback `onLoad`.

```
PageContainer container;

// Load synchronously and receive the callback after loading.
var handle = container.Push("FooPage", true, loadAsync: false, onLoad: x =>
{
    // Initialize page. (Called in the same frame as Push)
    x.page.Setup();
});

// Wait for the end of the transition animation.
yield return handle;
```

Chú ý nếu bạn sử dụng `AddressableAssetLoader` và tải đồng bộ, thì bạn cần sử dụng `Addressable` phiên bản **1.17.4 hoặc cao hơn**.

Ngoài ra, đây là một số cảnh báo về hiệu năng khi sử dụng `Addressable` `Addressable's` specifications (trong trường hợp sử dụng `Addressable` cùng với tải đồng bộ).

T?i tr??c

`Page` và `modal` chỉ được tải sau khi một chuyển tiếp màn hình được gọi. Khi tải một tài nguyên lớn nó có thể mất nhiều thời gian làm chuyển tiếp không còn mượt.

Trong trường hợp đó, việc tải trước tài nguyên là rất hữu ích.

Dưới đây là ví dụ tải trước với `PageContainer`.

```
const string pageName = "FooPage";
PageContainer container;

// Preload FooPage.
var preloadHandle = container.Preload(pageName);

// Wait for the end of the preloading.
yield return preloadHandle;

// Transition smoothly because FooPage has been preloaded.
container.Push(pageName, true);

// Release the preloaded FooPage.
container.ReleasePreloaded(pageName);
```

Xem thêm `HomePage` in demo để biết cách triển khai chi tiết.

Khi `Home` được khởi tạo. `Shop` cũng được load và destroy cùng lúc.

Các tính năng khác

?óng nhi?u màn hình cùng lúc

Trong `PageContainer` và `ModalContainer`, bạn có thể đóng nhiều màn hình cùng lúc. Để làm vậy hãy chỉ định số màn hình cần đóng trong tham số thứ 2 của `PageContainer.Pop()` hoặc `ModalContainer.Pop()`.

```
PageContainer pageContainer;  
pageContainer.Pop(true, 2);  
  
ModalContainer modalContainer;  
modalContainer.Pop(true, 2);
```

Bạn cũng có thể chỉ định `PageID` hoặc `ModalID`. `PageID` và `ModalID` có thể được lấy bằng cách sử dụng hàm callback `onLoad` của `Push()` như sau

```
PageContainer pageContainer;  
pageContainer.Push("fooPage", true, onLoad: x =>  
{  
    var pageId = x.pageId;  
});  
  
ModalContainer modalContainer;  
modalContainer.Push("fooModal", true, onLoad: x =>  
{  
    var modalId = x.modalId;  
});
```

Ngoài ra bạn có thể chỉ định bất kì ID nào bằng cách chỉ định tham số `pageID` hoặc `modalID` của hàm `Push()`

```
PageContainer pageContainer;  
pageContainer.Push("fooPage", true, pageID: "MyPageID");  
  
ModalContainer modalContainer;  
modalContainer.Push("fooModal", true, modalID: "MyModalID");
```

Đối với các page hoặc modal bị bỏ qua khi đóng nhiều màn hình. Các sự kiện vòng đời trước và sau khi chuyển tiếp sẽ không được gọi. Chỉ sự kiện trước khi destroy được gọi.

Trong `PageContainer`, animation của các page bị bỏ qua sẽ không chạy.

Trong `ModalContainer`, animation khi đóng các modal bị bỏ qua sẽ được phát đồng thời.

Không stack page trong lịch sử?

Đó là một số page mà bạn không muốn quay lại, ví dụ như màn hình loading.

Trong trường hợp này, bạn có thể chặn page stack trong lịch sử bằng cách khai báo `false` cho tham số `stack` trong hàm `PageContainer.Push()`.

Phiên bản này của page sẽ bị huỷ sau khi chuyển sang page tiếp theo nên do đó bị bỏ qua khi quay trở lại.

```
PageContainer container;  
  
// Transition to FooPage without stacking it in the history.  
yield return container.Push("FooPage", true, stack: false);
```

```
// Transition to BarPage, and FooPage is disposed.
yield return container.Push("BarPage", true);

// When Pop, it does not back to FooPage, but to the page before it.
yield return container.Pop(true);
```

Xem thêm TopPage in demo để biết cách triển khai chi tiết.
Chuyển sang page loading mà không cần stacking.

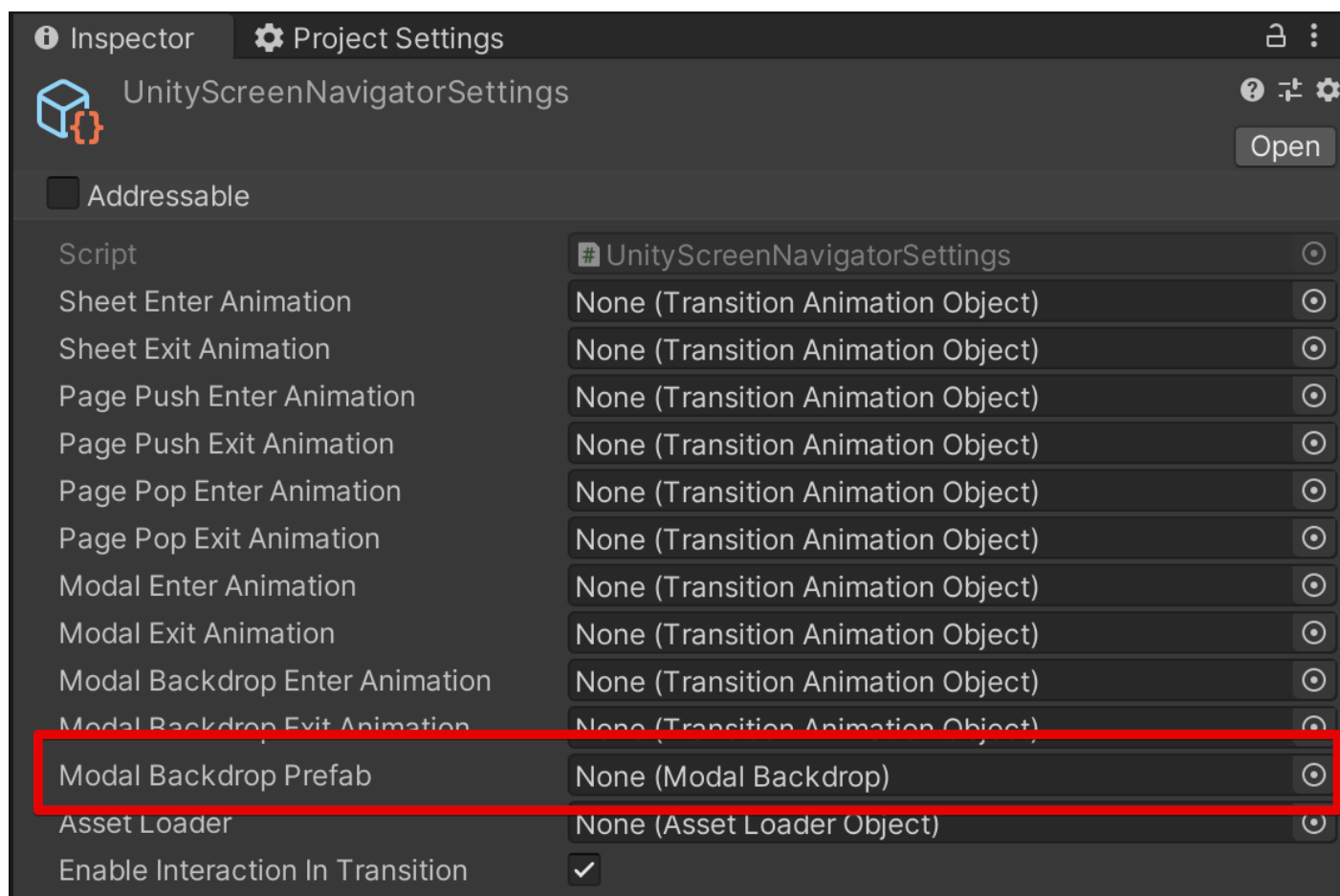
Thay đổi phong nền của modal

Theo mặc định, phong nền là một màn hình đen trong suốt. Bạn có thể thay đổi nó trong settings.

Để thay đổi nó bạn cần thêm component `ModalBackdrop` cho phong nền của modal và kéo nó thành Prefab.

Sau đó, gán prefab này như một phong nền của modal.

Để thay đổi phong nền của cả ứng dụng, hãy gán nó vào `ModalBackdropPrefab` trong `UnityScreenNavigatorSettings`.



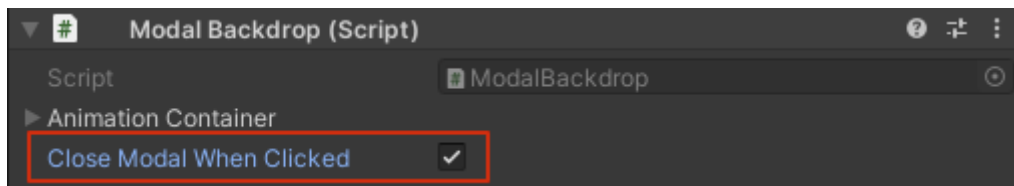
Bạn cũng có thể đặt phong nền cho mỗi `ModalContainer` bằng cách gán prefab vào `OverrideBackdropPrefab` của mỗi `ModalContainer`.

Cách modal hiển thị bằng cách click vào phong nền.

Mặc định, phong nền không thể click.

Nếu bạn muốn đóng modal khi click vào phông nền, đầu tiên hãy tạo một phông nền mới thay cho phông nền mặc định theo các bước trên.

Sau đó, check `CloseModalWhenClicked` trong component `ModalBackdrop`.

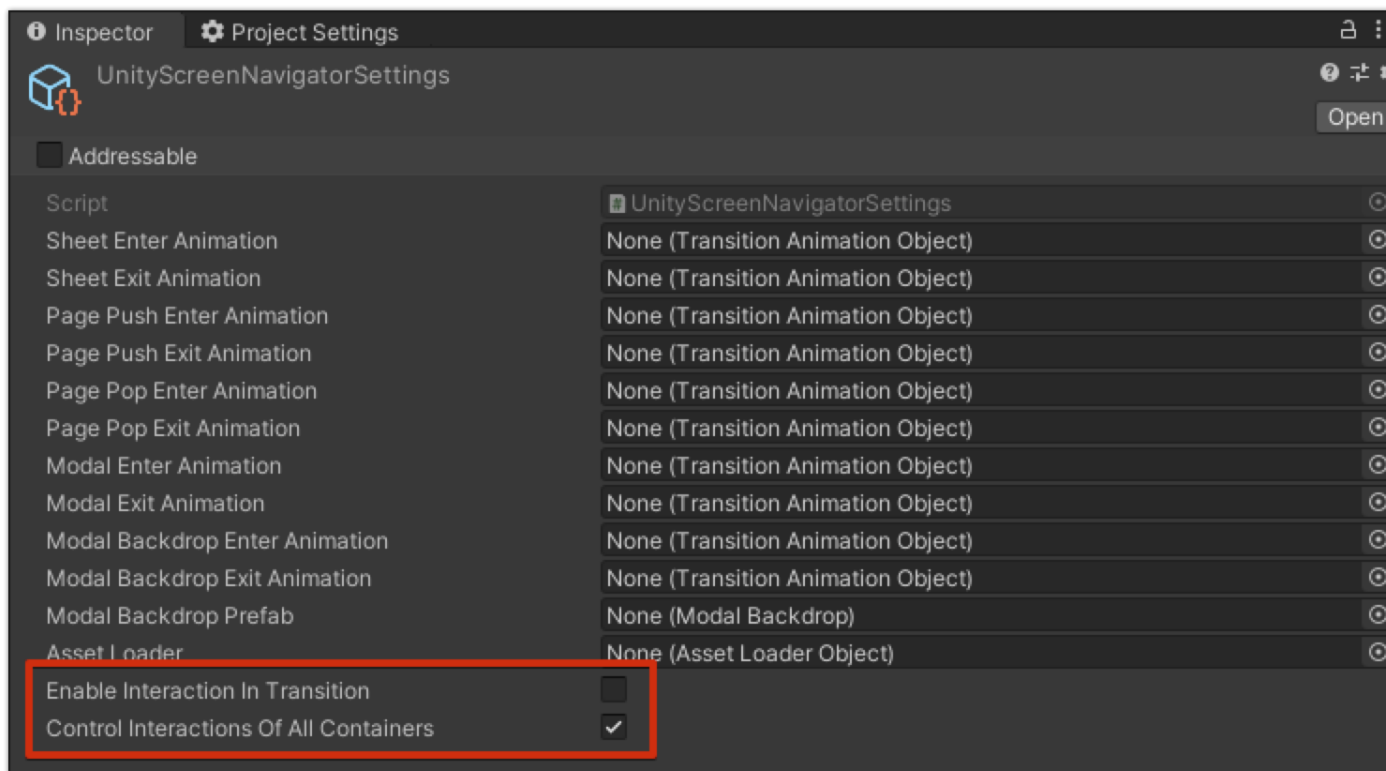


Cho phép tương tác khi đang chuyển màn hình.

Từ khi bắt đầu cho đến khi kết thúc chuyển màn hình, mọi tương tác đều bị vô hiệu hoá.

Bạn có thể thay đổi settings này bằng cách thay đổi thuộc tính `EnableInteractionInTransition` và `ControlInteractionsOfAllContainers` của `UnityScreenNavigatorSettings`. Mặc định, `EnableInteractionInTransition` là `false` và `ControlInteractionsOfAllContainers` là `true`.

Để cho phép tương tác khi chuyển màn hình, set `EnableInteractionInTransition` thành `true`. Và nếu bạn muốn vô hiệu hoá chỉ với container hiện tại đang chuyển, hãy giữ nguyên `EnableInteractionInTransition` là `false` và set `ControlInteractionsOfAllContainers` thành `false`.

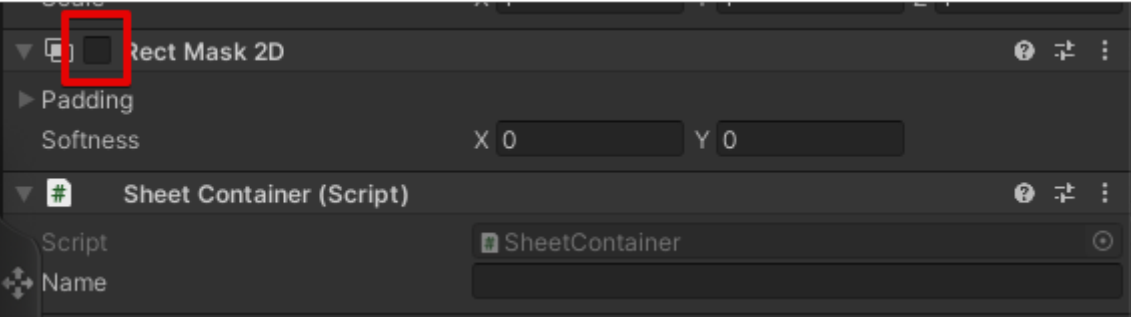


Tuy nhiên, không thể chuyển sang một màn hình khác khi đang chuyển trong một container. Do đó hãy kiểm soát thời gian chuyển đổi một cách thích hợp nếu bạn bật tương tác.

Tạo mask cho container

Mặc định, một phần của màn hình bên ngoài container bị che đi.

Nếu bạn muốn hiển thị màn hình bên ngoài container disable component `RectMask2D` của `GameObject` container.



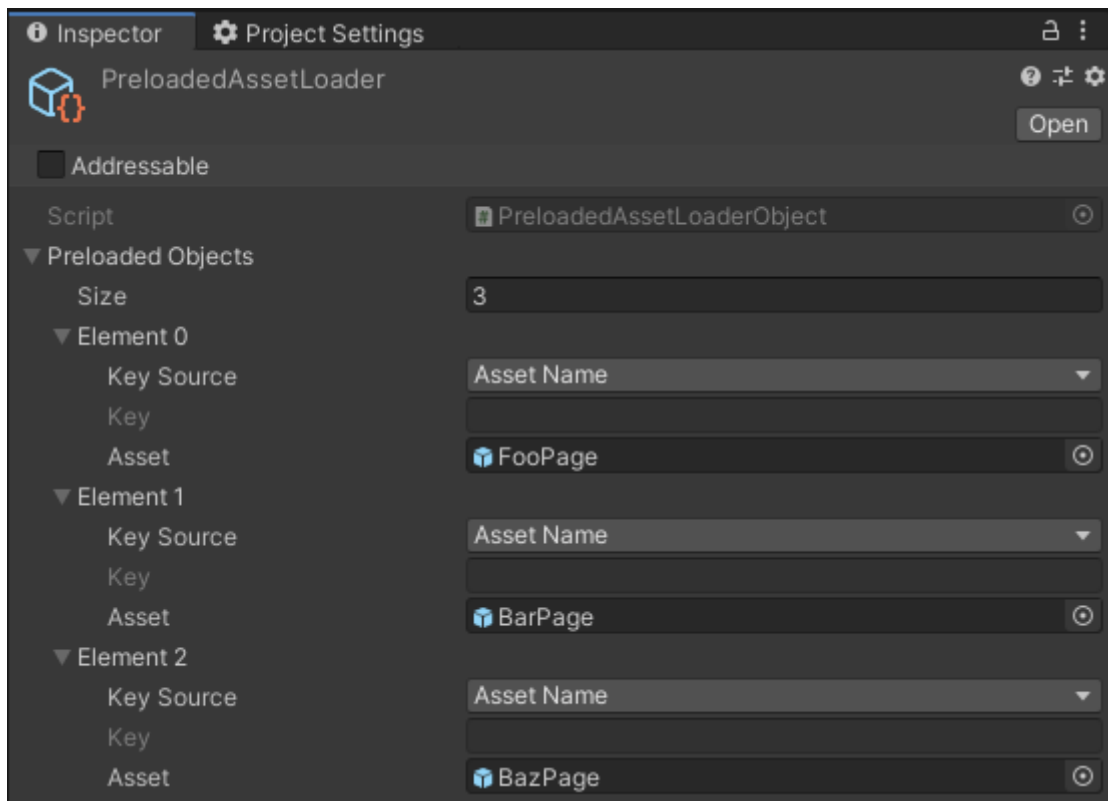
Lấy thông tin của animation đang chạy

Bạn có thể lấy thông tin của animation đang chạy từ các thuộc tính của lớp `Page`, `Modal`, `Sheet`.

Thuộc tính	Mô tả
<code>IsTransitioning</code>	Có đang trong quá trình chuyển màn hình hay không.
<code>TransitionAnimationType</code>	Loại animation, nếu không có hiệu ứng chuyển sẽ trả về null
<code>TransitionAnimationProgress</code>	Tiến trình của animation.
<code>TransitionAnimationProgressChanged</code>	Sự kiện khi tiến trình của animation thay đổi.

Sử dụng Prefab đã load khi tải màn hình.

`PreloadedAssetLoaderObject` cho phép bạn tải trực tiếp các prefab đã được tải trước thay vì sử dụng `Resource` hoặc `Addressable` khi tải các màn hình. Bạn có thể sử dụng bằng cách nhập key và prefab vào `ScriptableObject` tạo từ `Assets > Create > Resource Loader > Preloaded Asset Loader`.



Tôi cũng đã cung cấp triển khai `PreloaderAssetLoader` cho runtime.

Thay ??i hành vi c?a phong n?n.

Bạn có thể thay đổi hành vi của phong nền bằng cách thay đổi `BackdropStrategy` từ `ModalContainer`.

Giá trị	Mô tả
Generate Per Modal	Tạo một phong nền cho mỗi modal.
Only First Backdrop	Tạo một phong nền cho modal đầu tiên và không thêm phong nền cho bất kỳ modal nào tiếp theo.
Change Order Before Animation	Thay đổi thứ tự render của phong nền đầu tiên được tạo và sử dụng lại nó khi tạo modal thứ 2. (Điều chỉnh thứ tự render trước animation)
Change Order After Animation	Thay đổi thứ tự render của phong nền đầu tiên được tạo và sử dụng lại nó khi tạo modal thứ 2. (Điều chỉnh thứ tự render sau animation)

FAQ

Làm sao ?? t?o screen v?i Scene thay vì Prefab?

Bạn có thể tải màn hình được đặt trong Scene bằng cách triển khai `AssetLoader`. Triển khai interface `IAssetLoader` để tải Scene chứa màn hình được yêu cầu và trả về `GameObject` của màn hình. Xem Thay đổi phương thức load màn hình để biết thêm chi tiết.

Làm sao ?? tách biệt view và logic?

Tham khảo blog sau (viết bằng tiếng Nhật)

<https://light11.hatenadiary.com/entry/2022/01/11/193925>.

Làm sao ?? chuyển data giữa các màn hình?

Đầu tiên, trong một ví dụ, data được chuyển đến màn hình khi demo scene được tải xong.

[UnityScreenNavigator/Assets/Demo/Scripts/CharacterModal.cs](#)

Tuy nhiên, có nhiều cách để truyền dữ liệu. Ví dụ có thể bạn muốn sử dụng một DI Container (Dependency Injection) để set data. Do đó thư viện này không triển khai hoặc áp đặt một cách cụ thể nào?

Làm sao ?? tái sử dụng page và modal?

Đóng một page hoặc modal sẽ ngay lập tức destroy chúng và **không thể tái sử dụng**.

Bản chất của mong muốn tái sử dụng có thể chia thành 2 loại sau:

- Không muốn tải lại tài nguyên mỗi lần.
- Muốn duy trì trạng thái của màn hình.

Trong trường hợp này, vấn đề thời gian tải có thể được giải quyết bằng tải trước. Đối với việc duy trì trạng thái, để dễ bảo trì, trạng thái và giao diện nên được tách rời để có thể được tái tạo.

Ngoài ra, nhìn chung, từ quan điểm người dùng, chuyển đổi màn hình kiểu tab là kiểu nên giữ nguyên trạng thái. Trong thư viện này cũng vậy, trạng thái luôn được bảo toàn giữa các lần chuyển đổi Sheet để triển khai các tab. Xem tạo sheet để biết thêm chi tiết.

Và cuối cùng nếu tái sử dụng, người dùng có thể sẽ cần tự quản lý vòng đời. Nói cách khác, khi không cần thiết nữa người dùng phải gọi hàm Cleanup để hủy các instance và dọn dẹp bộ nhớ.

Giấy phép

Phần mềm này được phát hành theo giấy phép MIT License.

Bạn có thể tự do sử dụng trong phạm vi của giấy phép.

Tuy nhiên, các thông báo bản quyền và giấy phép sau đây phải được bao gồm khi sử dụng.

<https://github.com/Haruma-K/UnityScreenNavigator/blob/master/LICENSE.md>

Revision #2

Created 2025-06-09 10:13:28 UTC by ThanhDV

Updated 2025-06-22 06:33:28 UTC by ThanhDV