PopupFlow SDK – Queue-Based Popup SDK for Unity

Overview

PopupFlow is a **centralized, queue-based popup system** for Unity projects. It is designed to:

- Show clean, animated popups
- Handle multiple popup requests safely (one at a time)
- Provide flexible callbacks (Confirm, Cancel, Show, Hide)
- Avoid UI conflicts using a queue
- Work globally using a singleton pattern

This SDK is suitable for **error popups, retry dialogs, confirmations, alerts, and system messages**.

Architecture Summary

```
PopupFlow (Manager / Singleton)
|
├── PopupFlowData (Data Model)
|
└── PopupFlowUi (UI Controller)
    ├── Animations (DOTween)
    ├── Button bindings
    └── Canvas handling
```

---

1. PopupFlowData (Data Model)

Purpose

`PopupFlowData` is a **pure data container** that defines:

- Popup content
- Button labels
- Callback actions
- Hide behavior

It is **UI-agnostic** and safe to clone.

Fields

Field	Type	Description
Title	string	Popup title text
Description	string	Popup body message
ConfirmButtonText	string	Confirm button label
CancelButtonText	string	Cancel button label
OnConfirm	Action	Called when confirm button is clicked
OnCancel	Action	Called when cancel button is clicked
OnShow	Action	Called when popup is shown
OnHide	Action	Called when popup is hidden

HideOnConfirm	bool	Auto-hide after confirm (default true)
HideOnCancel	bool	Auto-hide after cancel (default true)

Clone Method

PopupFlowData Clone()

Used internally to prevent mutation of default popup data.

---

2. PopupFlowUi (UI Controller)

Purpose

Handles **visual presentation and user interaction**:

- Applies popup data to UI
- Binds button callbacks
- Plays show/hide animations
- Notifies PopupFlow when popup closes

Responsibilities

- ❌ No business logic
- ❌ No queue handling
- ✅ Only UI and animation

Key Components

Component	Purpose
CanvasGroup	Fade in/out

RectTransform (popupRoot)	Scale animation
TMP_Text	Title & description
Button	Confirm & Cancel

Public API

Show Popup

public void Show(PopupFlowData data)

- Applies data
- Registers button callbacks
- Plays show animation
- Invokes `OnShow`

Hide Popup

public void Hide()

- Plays hide animation
- Invokes `OnHide`
- Triggers `OnPopupHidden` event

Animation Details

- Fade: `CanvasGroup.DOFade`
- Scale: `RectTransform.DOScale`
- Duration: `0.3s`
- Ease:
  - Show: `Ease.OutBack`
  - Hide: `Ease.InBack`

⚠ Requires **DOTween**

---

3. PopupFlow (Popup Manager)

Purpose

Acts as the **central controller** for all popups:

- Singleton based
- Queue supported
- Default value handling
- Multiple overloads for ease of use

Singleton Behavior

PopupFlow.Instance

- Automatically persists using `DontDestroyOnLoad`
- Prevents duplicate instances

Popup Queue System

Why Queue?

To avoid:

- Multiple popups overlapping
- UI input conflicts
- Lost popup calls

How It Works

1. Popup requested
2. If popup active → enqueue
3. If not active → show immediately
4. When popup hides → dequeue next

Queue<PopupFlowData> popupQueue

Default Popup Data

`DefaultData` is used as a **base template**.

If not assigned, it auto-initializes:

Title = "Alert"

Description = "Something happened"

ConfirmButtonText = "OK"

CancelButtonText = "Cancel"

All popups clone this data and override selectively.

Main Show Method (Core API)

```
PopupFlow.Instance.Show(

    title,

    description,

    confirmButton,

    cancelButton,

    onConfirm,

    onCancel,

    onShow,

    onHide,

    hideOnConfirm,

    hideOnCancel
);
```

This is the **master method** used internally by all overloads.

Common Usage Examples

Simple Alert

```
PopupFlow.Instance.Show(
    "Warning",
    "Internet connection lost"
);
```

Confirm Dialog

```
PopupFlow.Instance.Show(
    "Exit Game",
    "Are you sure?",
    "Yes",
    "No",
    onConfirm: QuitGame
);
```

Retry Popup (Do not hide on confirm)

```
PopupFlow.Instance.Show(
    title: "Error",
    description: "Request failed",
    confirmButton: "Retry",
    onConfirm: RetryRequest,
    hideOnConfirm: false
);
```

Full Control

```
PopupFlow.Instance.Show(
    "Upload",
    "Upload completed",
    "OK",
    null,
    onConfirm: OnOk,
    onCancel: null,
    onShow: () => Debug.Log("Shown"),
    onHide: () => Debug.Log("Hidden")
);
```

Cancel Button Visibility Rules

Cancel button is shown **only if**:

- `CancelButtonText` is provided OR
- `OnCancel` callback exists

Otherwise, it is hidden automatically.

Force Hide

PopupFlow.Instance.Hide();

Immediately hides the current popup (useful for scene changes).

EventSystem Auto-Creation

PopupFlow ensures UI input always works:

EnsureEventSystemExists()

If missing, it automatically creates:

- `EventSystem`
- `StandaloneInputModule`

Design Principles

- ✅ Separation of concerns
- ✅ Queue safety
- ✅ Callback driven
- ✅ Reusable & scalable
- ✅ Game-pause friendly

Recommended Use Cases

- API retry dialogs
- Network error popups
- Confirmation prompts
- System alerts

- Game flow blocking dialogs

Dependencies

- Unity UI
- TextMeshPro
- DOTween

Notes for Developers

- Do not directly manipulate `PopupFlowUi`
- Always use `PopupFlow.Instance.Show()`
- Customize visuals only inside `PopupFlowUi`
- Extend `PopupFlowData` if needed

PopupFlow SDK is designed to be clean, safe, and production-ready for scalable Unity projects.

Author: Abhishek Sahu