```csharp
using System;
using UnityEngine;
using UnityEngine.UI;

/// <summary>
/// GuideHandler is assigned to its respective prefab, allowing for transitions between windows and tabs.
/// </summary>
public class GuideHandler : MonoBehaviour
{
    /// <summary>
    /// The button color of a selected and unselected tab respectively.
    /// </summary>
    [SerializeField]
    Color selectedTabColor, unselectedTabColor;

    /// <summary>
    /// The button backgrounds of each window.
    /// </summary>
    [SerializeField]
    Image welcomeWindow, logicGatesWindow, controlsWindow;

    /// <summary>
    /// The button backgrounds of the tabs within each window.
    /// </summary>
    [SerializeField]
    Image[] welcomeTabs, logicGatesTabs, controlsTabs;

    /// <summary>
    /// The GameObjects that contains all tabs of each window respectively.
    /// </summary>
    [SerializeField]
    GameObject welcomeTabsParent, logicGatesTabsParent, controlsTabsParent;

    /// <summary>
    /// The view areas of the tabs within each window.
    /// </summary>
    [SerializeField]
    GameObject[] welcomeTabsViews, logicGatesTabsViews, controlsTabsViews;

    /// <summary>
    /// The background button of the current window that is viewable.
    /// </summary>
    private Image currentWindow;

    /// <summary>
    /// Displays the index of the currently opened tab for each window.
    /// </summary>
    private int currentWelcomeTab = 0, currentLogicGatesTab = 0, currentControlsTab = 0;

    // Initializes the default window as the welcome window.
    private void Start() { currentWindow = welcomeWindow; }

    /// <summary>
    /// Switches to a new tab for the current window in use.
    /// </summary>
    /// <param name="newTab">The index of the tab to switch to.</param>
    public void UpdateTab(int newTab)
    {
        GameObject currentView, newView; // The view areas to turn off and on respectively.
        Image currentButton, newButton; // The button backgrounds to color unselected and selected respectively.

        // Populates initialized variables based on current window.
        if (currentWindow == welcomeWindow)
        {
            if (newTab == currentWelcomeTab) return;

            currentButton = welcomeTabs[currentWelcomeTab];
            newButton = welcomeTabs[newTab];
            currentView = welcomeTabsViews[currentWelcomeTab];
            newView = welcomeTabsViews[newTab];
            currentWelcomeTab = newTab;
        }

        else if (currentWindow == logicGatesWindow)
        {
            if (newTab == currentLogicGatesTab) return;

            currentButton = logicGatesTabs[currentLogicGatesTab];
            newButton = logicGatesTabs[newTab];
            currentView = logicGatesTabsViews[currentLogicGatesTab];
            newView = logicGatesTabsViews[newTab];
            currentLogicGatesTab = newTab;
        }

        else if (currentWindow == controlsWindow)
        {
            if (newTab == currentControlsTab) return;

            currentButton = controlsTabs[currentControlsTab];
            newButton = controlsTabs[newTab];
            currentView = controlsTabsViews[currentControlsTab];
            newView = controlsTabsViews[newTab];
            currentControlsTab = newTab;
        }

        // Incorrent window currently in use
        else throw new Exception("Invalid current window.");

        // Updates the obtained values
        currentButton.color = unselectedTabColor;
        newButton.color = selectedTabColor;
        currentView.SetActive(false);
        newView.SetActive(true);
    }

    /// <summary>
    /// Switches to a new window.
    /// </summary>
    /// <param name="newWindow">The new window to switch to.</param>
    public void UpdateWindow(Image newWindow)
    {
        if (currentWindow == newWindow) return;

        // Updates the button colors.
        currentWindow.color = unselectedTabColor;
        newWindow.color = selectedTabColor;

        // Makes the current window invisible.
        if (currentWindow == welcomeWindow) welcomeTabsParent.SetActive(false);

        else if (currentWindow == logicGatesWindow) logicGatesTabsParent.SetActive(false);

        else if (currentWindow == controlsWindow) controlsTabsParent.SetActive(false);

        else throw new Exception("Invalid current window.");

        // Makes the new window visible.
        if (newWindow == welcomeWindow) welcomeTabsParent.SetActive(true);

        else if (newWindow == logicGatesWindow) logicGatesTabsParent.SetActive(true);

        else if (newWindow == controlsWindow) controlsTabsParent.SetActive(true);

        else throw new Exception("Invalid new window.");

        currentWindow = newWindow;
    }
}
```