WLS BUTTON MANAGER

Documentation

WILLOW LAKE STUDIO

Asheville, NC

Introduction

WLS Button Manager allows you to accept Tab/Shift+Tab key inputs to navigate through standard Unity UI Buttons in a scene. You can assign which button has focus at the start of the scene. The current button shows the assigned Selected Color on focus and mouse hover. It allows the navigation through the list of buttons in the scene to "wrap around" from the last to the first and vice versa. It does not override Unity's default Horizontal/Vertical axes input to navigate through the buttons. Submit button input (default: Space/Enter) will trigger the OnClick event of the focused button.

This asset package consists of two scripts:

- **WLSButton**. Attached to each button you want to be able to tab through. Assign the default button in the scene.
- **WLSButtonManager**. Attached to a game object in the hierarchy and controls the input.

This asset package consists of two prefabs:

- **WLSButton**. This prefab is a standard Unity UI Button with a higher contrast "Selected Color" assigned to make it easier to tell which button has focus. It has the WLSButton script attached and the “WLSButton” tag assigned.
- **WLSButtonManager**. This prefab is an empty game object with the WLSButtonManager script attached and the “WLSButtonManager” tag assigned.

This asset package includes a sample scene in **WLS Button Manager > Assets > Scenes**. Use this to examine a functional demo setup.

Please note: All five buttons in the demo scene have the same **Demo** script attached and call the **Demo.OnClick** function to output a **Debug.Log** message when each button is clicked. Your implementation will need to have separate scripts or at least functions assigned to each button to execute their own code when clicked.

Simple Implementation

Since this uses standard Unity UI buttons, you will need a Canvas and Event System in your scene.

1. Open the **WLS Button Manager > Assets > Prefabs** folder in the Project view.
2. Drag the **WLSButtonManager** prefab into the Canvas in the Hierarchy.
3. Drag the **WLSButton** prefab into the Canvas in the Hierarchy. Repeat for each button you want to be able to tab through.
4. Rename and position the buttons. Change their appearance to your liking. This script puts an emphasis on the SelectedColor property on the standard Button script.
5. On each button, assign the following in the Inspector:
    - **Button**. Assign the OnClick() event as you normally would in the standard Unity UI Button script.
    - **WLS Button.Has Focus**. The default button when the scene loads. Only one button should have this option checked.

Custom Implementation

Along with the required steps in the Simple Implementation section, feel free to expand the scripts further. For example:

Accepting input from other axes:

The WLSButtonManager script is currently hard-coded to get input from the standard keyboard Tab and Shift+Tab keys. The standard Unity UI Button already accepts Horizontal/Vertical axes inputs (by default keyboard arrow keys) to navigate through UI elements and accepts "clicks" from the Submit axes as well as actual mouse clicks.

To add an axial input, duplicate and modify the **TabInput** function in the WLSButtonManager script. Input.GetKeyDown would need to be changed to Input.GetAxisRaw and treat the -1/1 input the same as the Shift+Tab/Tab keys are being treated.

To add a button or key input, duplicate and modify the **TabInput** function in the WLSButtonManager script, assigning your own keycode.

Highlighting the focused button:

This asset currently calls the SetSelectedGameObject() on the EventSystem when a button gets focus.

If you wanted to trigger another behavior, for example playing an effect, animation, or particle system on the button, you can create a custom function and call it in the following places:

- **WLSButton.Start**. If this button has focus, EventSystem.current.SetSelectedGameObject() is called.
- **WLSButtonManager.SelectButton**. If this is the next button in the list, EventSystem.current.SetSelectedGameObject() is called.

Eliminating repeated Tab key input:

Changing the *Input.GetKeyDown(KeyCode.Tab)* line in WLSButtonManager.cs to *Input.GetKeyUp(KeyCode.Tab)* will eliminate repeated Tab keystrokes.

Known Issues

- You can only have one WLS Button Manager component in a scene. More than one will obviously cause conflicts.
- This asset is designed for use in a single scene, where there is a finite list of button inputs from the start. It does not allow for buttons to be dynamically added or removed from the list.
- It is designed to be added separately to each scene, not as part of a DontDestroyOnLoad game manager type object that persists from one scene to the next. This would only work if the buttons you want to manage also persist from one scene to the next.
- A layered overlay menu, for example, with Pause panel, Settings panel, Tutorial panel, etc. will cause unexpected behaviors, as WLS Button Manager will try to navigate through all tagged buttons in the scene, currently visible or not.
- This has hard-coded the expected input of Tab and Shift+Tab. If you have these inputs assigned to another axis, both actions will be triggered.

Release Notes

V1.0

- Initial release

V1.1

- Combined WLSButtonHover.cs functions into WLSButton.cs
- Removed unused OnClick() function in WLSButton.cs
- Changed multiple .Select() calls to EventSystem.current.SetSelectedGameObject()

Help & Support

Watch the implementation demo in the video linked in the Asset Store.

Then email: james@willowlakestudio.com

Visit https://willowlakestudio.com/unity for more projects and demos.