Common Bound Action Bar

Use the Common Bound Action Bar to automatically display a list of button hints for your UI.

User Interfaces can map Input Actions to onscreen buttons. For example, an options menu with multiple tabs may use the left and right shoulder buttons on a gamepad to toggle between multiple tabs. Interactions such as these are highly contextual, therefore, CommonUI includes a widget called the **Common Bound Action Bar**, which shows all Input Actions in your currently active UI in a single, easily referenced location. Typically, this widget is placed across the bottom of the screen.

How the CommonBoundActionBar / NavBar Works

`UCommonBoundActionBar` updates on Tick. The final bar update is implemented in the function *UCommonBoundActionBar::HandleDeferredDisplayUpdate*. Any Input Actions that have the `bDisplayInActionBar` property set to `true` are gathered, sorted, then added to a list for display.

This update is bound to the **Action Router** delegate `UCommonUIActionRouterBase::OnBoundActionsUpdated_`, which fires during various node change points in CommonUI. Node changes occur whenever a widget activates or deactivates, so this is a perfect point to trace available action changes.

However, `UCommonUIActionRouterBase` is a local player subsystem, which means it relies on the local player for Ticking. This means that when games pause Ticking to display a menu, `UICommonBoundActionBar` will not update dynamically as the available CommonUI actions change, because it depends on the player's Tick process.

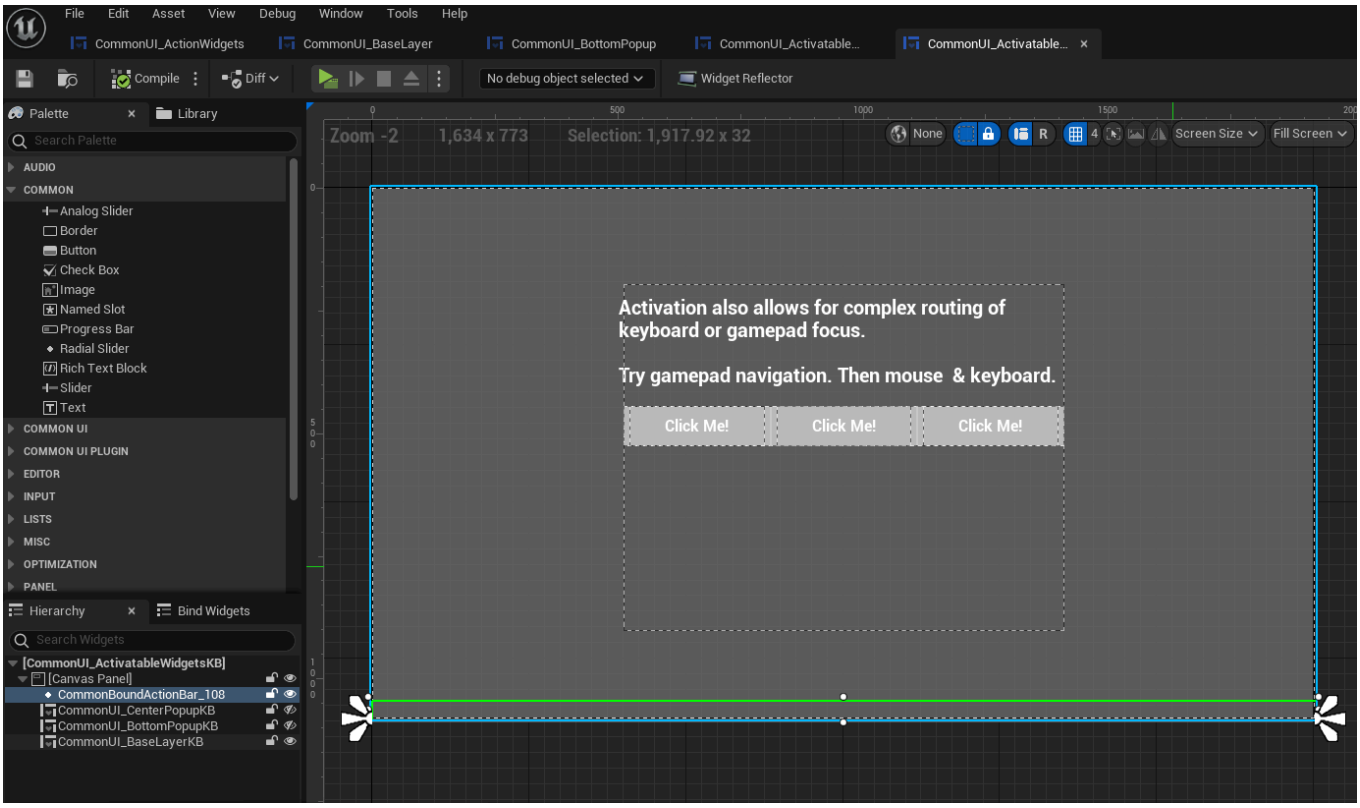
Not all actions are added to the action bar, nor do they have to be. `FBindUIActionArgs::bDisplayInActionBar` determines if an Input Action will get added to the action bar. This is exposed in Blueprints through the **Display In Action Bar** setting, and you can also access it with `UCommonUserWidget::bDisplayInActionBar` in C++.

> As a possible workaround, you can enable Ticking while paused for the Actor or local player who owns player interactions or your UI. You can also make subclasses of widgets and set those to be tickable when paused.

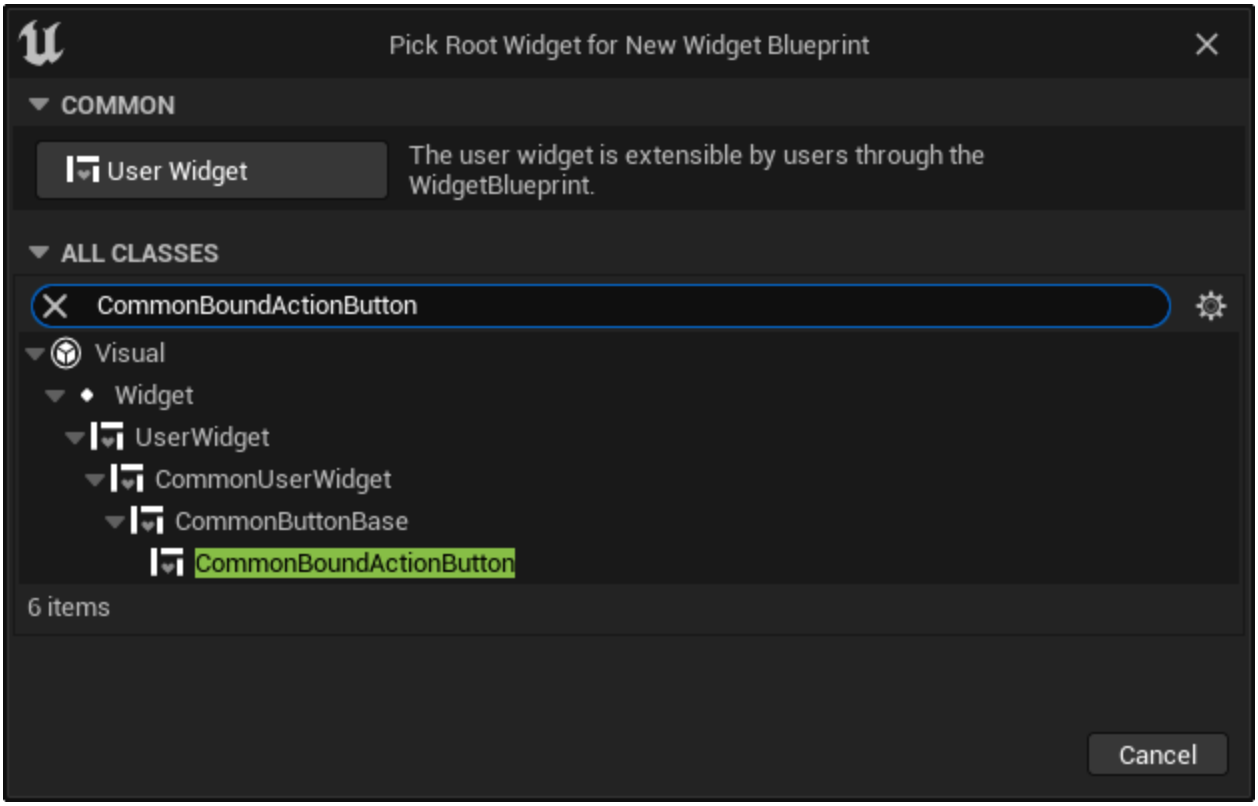
Implement the CommonBoundActionBar in Your UI

To set up the Common Bound Action Bar:

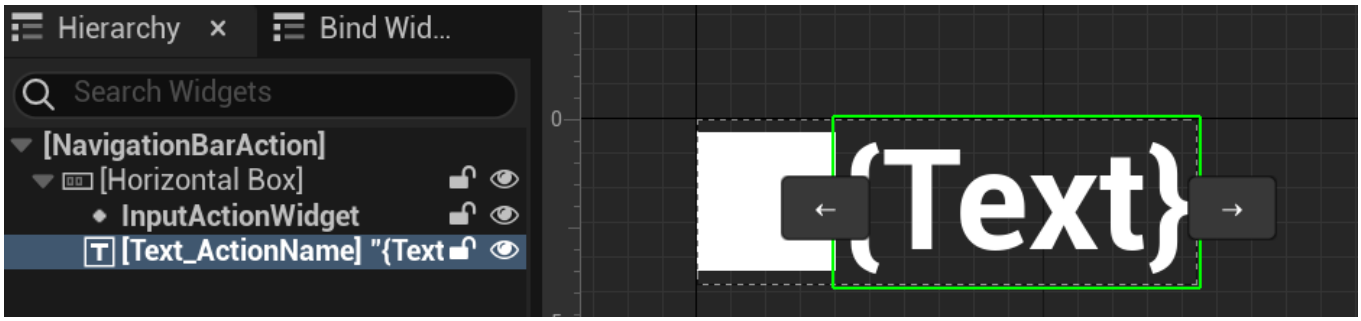
1. Add a CommonBoundActionBar to your Widget Blueprint. The Content Examples project anchors it to the bottom of the screen in **CommonUI_ActivatableWidgetsKB**. You can find this widget in **ExampleContent** > **UMG** > **CommonUI** > **ActivatableWidgetsKB**.



2. Create a class that derives from `UCommonBoundActionButton`. In the Content Examples project, this widget is named **NavigationBarAction**.



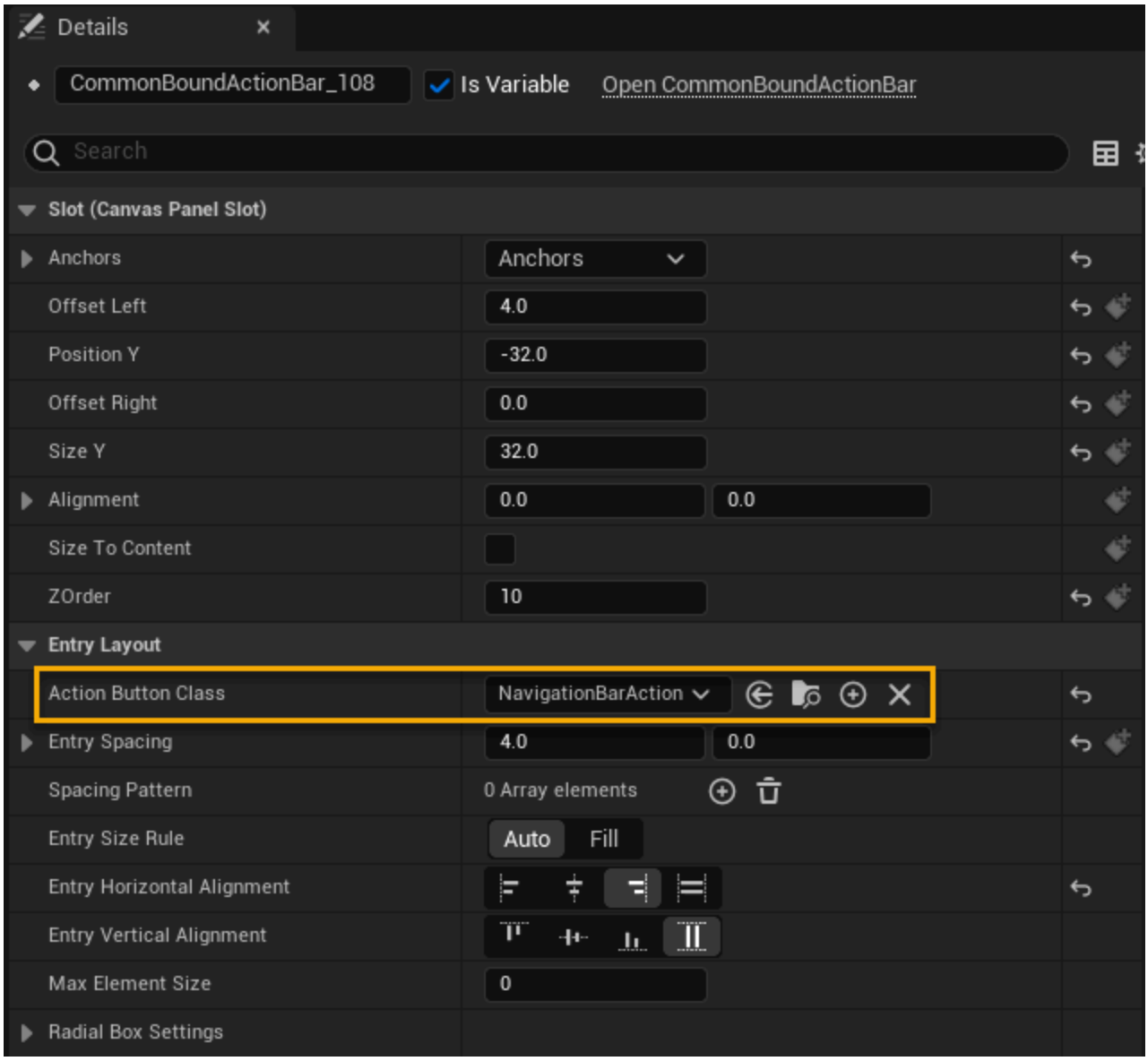
3. For a simple implementation, use a **Common Input Action** widget and a **Common Text Widget** in a horizontal box. The Common Input Action widget displays your button icons, while the Common Text Widget displays the friendly name for the Input Action.



4. Name your Common Text Widget "**Text_ActionName**". `UCommonBoundActionButton` binds the text widget to the InputAction's text based on this specific name.

> If you do not rename your Common Text Widget to "Text_ActionName", your Blueprint will fail to compile.

5. Add your CommonBoundActionButton to the **Action Button Class** for your CommonBoundActionBar.



6. Select the CommonUI widgets in your UI whose actions you want to display in the CommonBoundActionBar, then set **Display in Action Bar** to **true**. In C++, this is represented with `bDisplayInActionBar`.

> `bDisplayInActionBar` is a member of `UCommonUserWidget` and its derived classes, like `UCommonButtonBase`. Like Input Actions themselves, it is not available in base UMG's library of widgets.

7. Make sure that the widgets containing the Input Actions you want to display are **Activated**. This means that either the widgets themselves must be Activatable Widgets, or they must be children of an Activatable Widget.

Activatable Widgets start in a deactivated state by default. You can call `UCommonUserWidget::Activate` to manually activate these widgets, or you can use the **Auto Activate** setting (UCommonActivatableWidget::bAutoActivate) to make them self-Activate on being added to the Viewport.

When playing, assuming the widgets containing your actions are activated, the actions should display in the navigation bar.