UGUI Navigation Wizard Manual

Table of contents

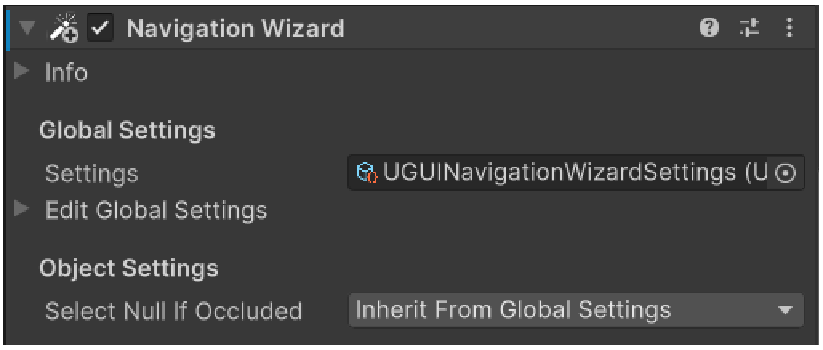
Requirements & Setup

Requirements

Unity 2021.3 or higher is required since that is the min version Unity allows for new assets in the store. However it may work just fine in older versions of Unity with minor changes.

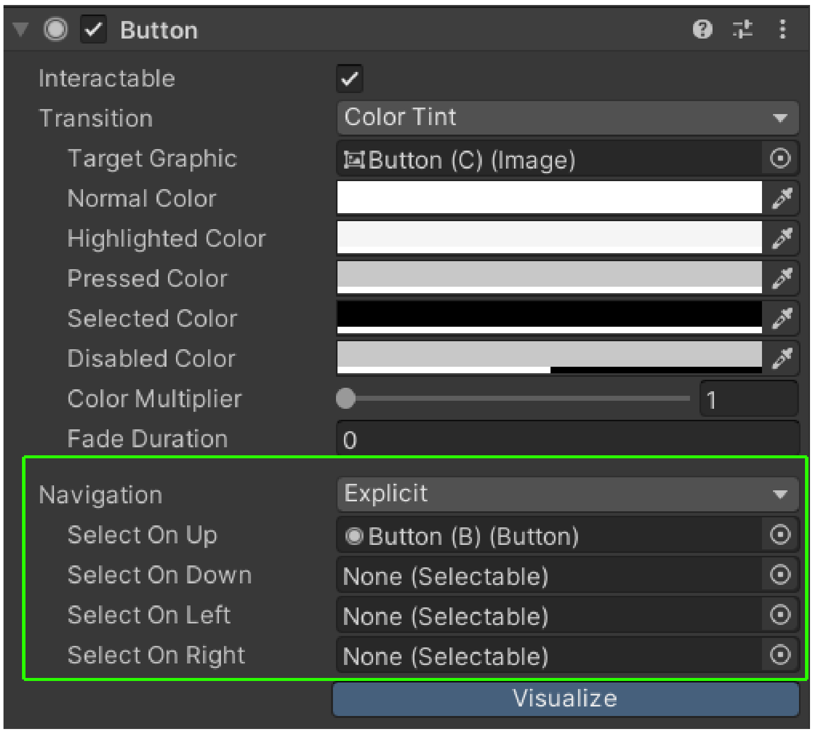
Overview

The navigation wizard works by adding a component (UGUI Navigation Wizard) to your selectables (buttons, inputs, ...). It sounds cumbersome to add yet another component to your many buttons? Well, there is a prefab that does it all automatically at runtime if you want.



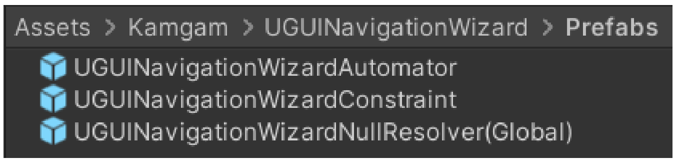
Navigation Wizard
Info
Global Settings
Settings: UGUINavigationWizardSettings (U
Edit Global Settings
Object Settings
Select Null If Occluded: Inherit From Global Settings

This component then takes control over the navigation section of your selectable.



Button
Interactable: ✓
Transition: Color Tint
Target Graphic: Button (C) (Image)
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier: 1
Fade Duration: 0
Navigation: Explicit
Select On Up: Button (B) (Button)
Select On Down: None (Selectable)
Select On Left: None (Selectable)
Select On Right: None (Selectable)
Visualize

Some features (like „Null Resolution“, „Constraints“ and „Automation“) will require you to add additional components to your scene (there are some pre-made prefabs):



Assets > Kamgam > UGUINavigationWizard > Prefabs
UGUINavigationWizardAutomator
UGUINavigationWizardConstraint
UGUINavigationWizardNullResolver(Global)

And that’s about it. Please read on in the „Getting Started“ section below.

Getting Started

If you want to try the wizard without already adding extra components to your UI then the AUTOMATOR (UGUI Navigation Wizard Automator) is the right tool for the job.

What does it do?

At runtime it will simply search (every frame) for new selectables and if it finds some then it will add a temporary „UGUI Navigation Wizard Component“ to it. That way you can already see the navigation changes and play around with it without adding anything permanently to your scene.

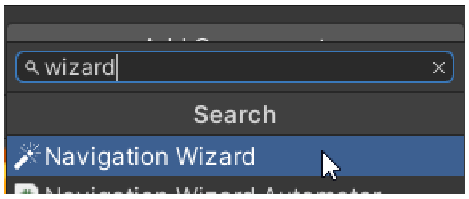
Won't that cause some hickups at first UI display if the automator adds all these components?

Yes, that's why I recommend to not use it once you have decided that the wizard works for you. Once you know it does okay in your project you can start adding the UGUI Navigation Wizard component to your selectables permanently.

HINT: The automator inspector has some handy buttons to do this for you.

How to add a navigation wizard to a button?

Easy, simply search for it in the „Add Component“ menu.

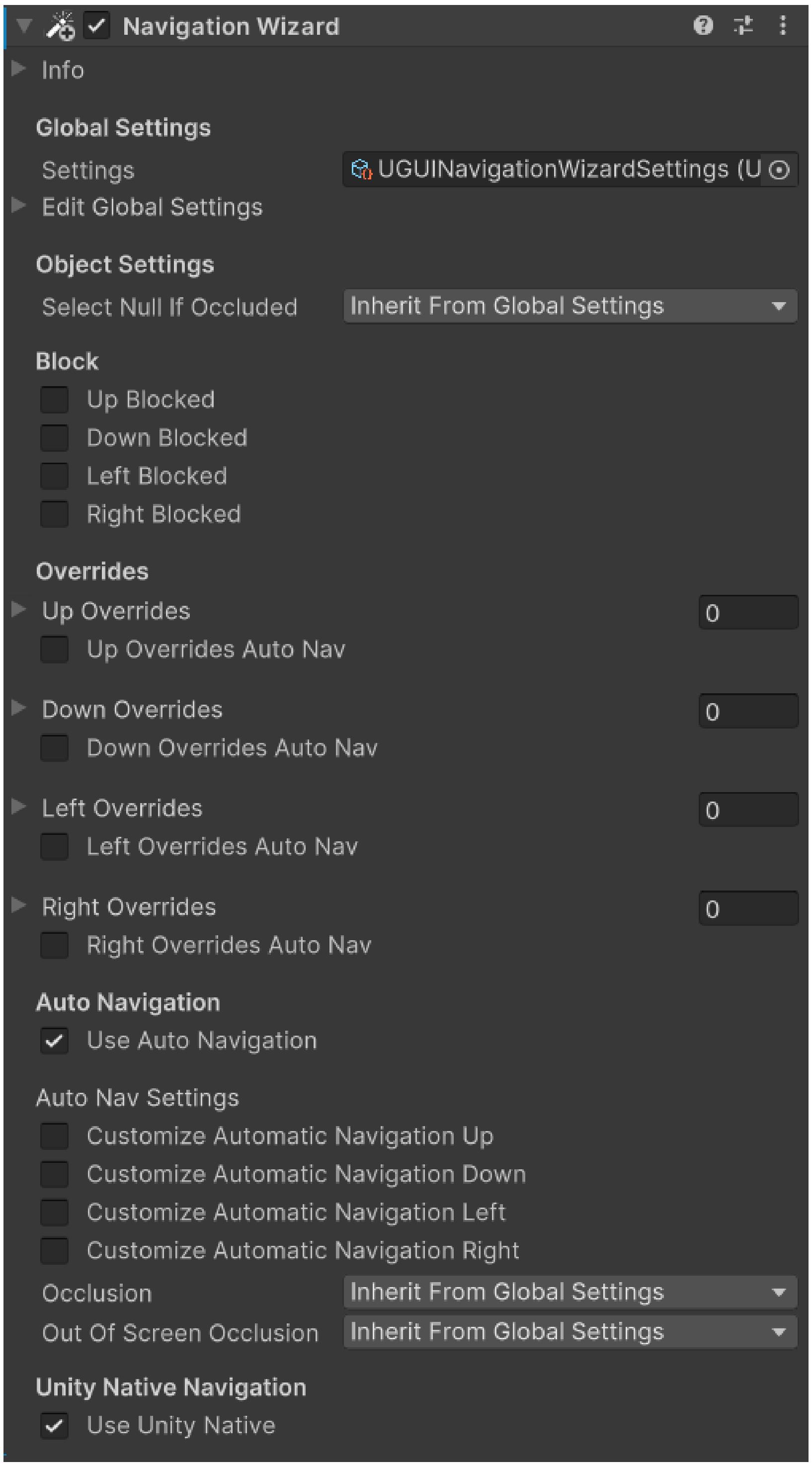


And that's it.

For more details on the navigation wizard please consult the „Navigation Wizard Options“ section below.

Navigation Wizard Options

How to add a navigation wizard to a button?
Easy, simply search for it in the „Add Component" menu.



Navigation Wizard

Info

Global Settings

Settings: UGUINavigationWizardSettings (U

Edit Global Settings

Object Settings

Select Null If Occluded: Inherit From Global Settings

Block

- [ ] Up Blocked
- [ ] Down Blocked
- [ ] Left Blocked
- [ ] Right Blocked

Overrides

Up Overrides: 0
- [ ] Up Overrides Auto Nav

Down Overrides: 0
- [ ] Down Overrides Auto Nav

Left Overrides: 0
- [ ] Left Overrides Auto Nav

Right Overrides: 0
- [ ] Right Overrides Auto Nav

Auto Navigation

- [x] Use Auto Navigation

Auto Nav Settings

- [ ] Customize Automatic Navigation Up
- [ ] Customize Automatic Navigation Down
- [ ] Customize Automatic Navigation Left
- [ ] Customize Automatic Navigation Right

Occlusion: Inherit From Global Settings

Out Of Screen Occlusion: Inherit From Global Settings

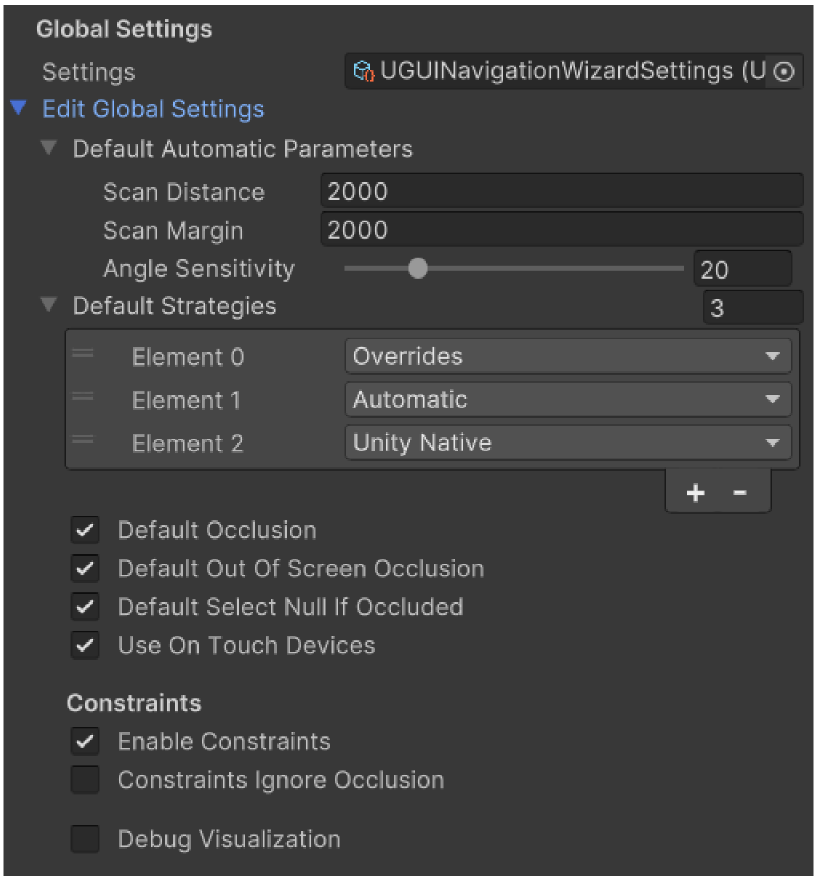
Unity Native Navigation

- [x] Use Unity Native

The navigation wizard has multiple sections. Below you will find a detailed description of each of them.

Global Settings

The base configuration is the same for all wizard components and it is stored in a Scriptable Object (SO) called UGUINavigationWizardSettings. You can see it linked in the „Settings“ field.



Global Settings

Settings: UGUINavigationWizardSettings (U

Edit Global Settings

Default Automatic Parameters

Scan Distance	2000
Scan Margin	2000
Angle Sensitivity	20

Default Strategies: 3

Element 0	Overrides
Element 1	Automatic
Element 2	Unity Native

- ✓ Default Occlusion
- ✓ Default Out Of Screen Occlusion
- ✓ Default Select Null If Occluded
- ✓ Use On Touch Devices

Constraints

- ✓ Enable Constraints
- ☐ Constraints Ignore Occlusion

- ☐ Debug Visualization

The „Edit Global Settings“ foldout is a shortcut for editing the most important settings. It saves you from having to open the settings SO every time you want to change something.

NOTICE: Changes made here will immediately effect ALL wizards in the project, hence the name „GLOBAL“ settings ;-).

For more details on the „Auomatic Parameters“ please refer to the „Auto Navigation“ section below. For now all you need to know is that these are the default values used for the auto navigation unless you customize them on the component (again please refer to the section below).

The default strategies are a list of ORDERED strategies the wizard will use to control the up, down, left and right navigation targets. First it will check if you have set some overrides on the local wizard component. If yes, then it will use these. If it finds a valid navigation target there then it will use that one. If it does not find a target then it will proceed with the next strategy in the list (Automatic) and if even that does not yield any valid selectable then it will finally fall back on Unitys native selection.

NOTICE: Modifying these here is not advisable since they are global.

HINT: You can enable/disable them on a per wizard basis via the „Use …“ checkboxes (more on that below).

Default Occlusion

Should occlusion detection be enabled. By default it is enabled. If you disable it then, well, no occlusion detection will be done.

Default Out Of Screen Occlusion

If enabled then objects outside the screen will not be selectable.

NOTICE: Disabling this will make any object outside the screen selectable. There is no normal occlusion detection performed outside the screen. Disable with care. However, it is useful for scroll views and similar ui elements that routinely have elements outside the screen.

HINT: There is a component called „UIGUI Navigation Wizard Set Out Of Screen Occlusion“ which you can use to override this property in all children. Place it on the viewport of your scroll view to disable out-of-screen occlusion for all elements inside the scroll view.

HINT: There is a „UGUI Scroll Into View On Select“ component which is useful for auto-scrolling scrol views if controlled by keyboard or controller.

Default Select Null If Occluded

If an object that is selected becomes occluded then we usually want to react to that. The default reaction is to set the selection to null. This has the nice side effect that once it was set to null the NullResolver will kick in and fix the selection. If you do not want this to happen then you can disable it here.

HNT: You can also set this on a per object basis.

Use On Touch Devices

On touch devices the selection navigation is usually irrelevant. You can completely disable the system on touch devices to save performance. Please only disable it if the profiler tells you this is the bottleneck and if you don’t plan to support mobile controllers.

Constraints Ignore Occlusion

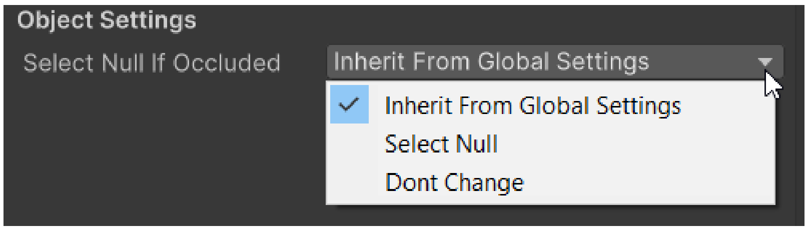
Here you can configure whether or not objects that are in the allowed list of any constraint should be selectable even if they are occluded. This is handy for some edge cases like tutorials where some ui may overlap with a button that actually should be selectable.

Debug Visualization

As the name suggests this enables some debug visualizations, primarily for the improved automatic navigation. It will show you some of the points used for distance calculations. If you zoom out really far it will also show the auto navigation bounding boxes.

Object Settings

These are settings that are on a per-object basis. This means you can customize them for each selectable.



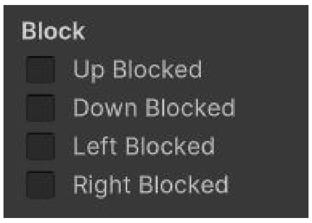
Select Null If Occluded

If an object that is selected becomes occluded then we usually want to react to that. The default reaction is to set the selection to null. This has the nice side effect that once it was set to null the NullResolver will kick in and fix the selection. If you do not want this to happen then you can disable it here.

You can also choose to inherit this setting from the global settings (default).

Blocking

Here you can block each navigation direction. Notice that if you block a direction none of the other navigation strategies will be applied (no override, no auto navigation).

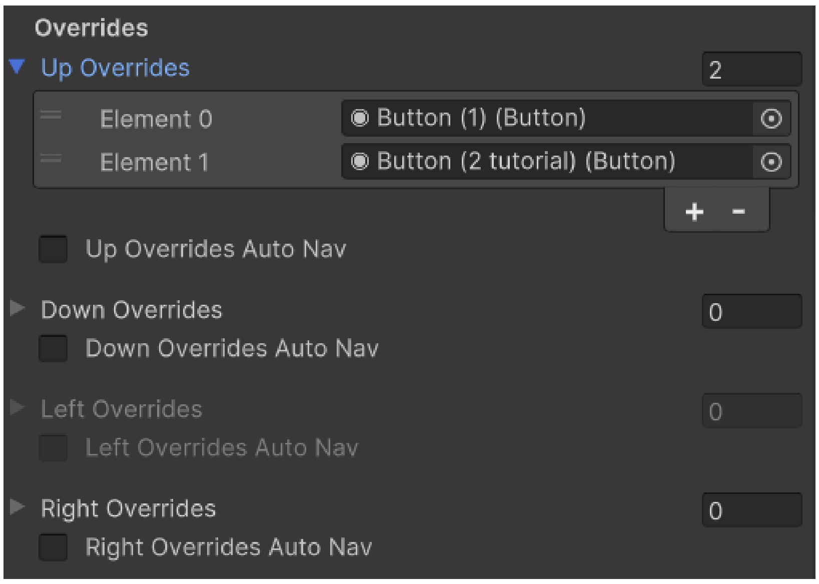


Overrides

Overrides allow you to mix explicit (overridden) and automatic navigation.

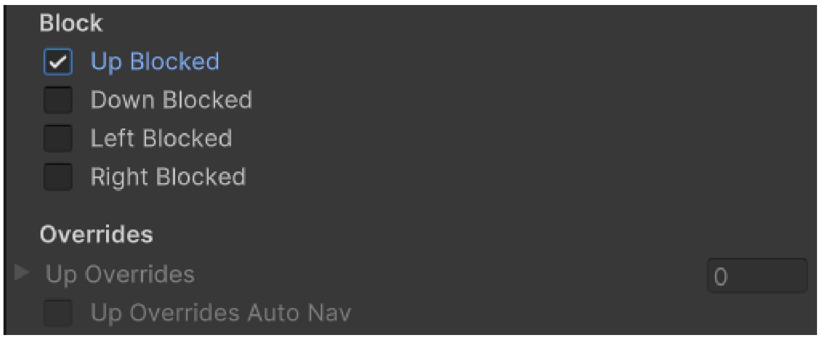
If you specify overrides for a navigation direction then these will take precedence over the automatic navigation. If you leave them empty or if all of the overrides in the list are invalid (not active, occluded, ..) then the automatic navigation will take over.

The overrides are checked in a top down order (element 0, 1, 2, ..) and the first that is valid will be chosen.



Overrides
Up Overrides 2
Element 0 Button (1) (Button)
Element 1 Button (2 tutorial) (Button)
Up Overrides Auto Nav
Down Overrides 0
Down Overrides Auto Nav
Left Overrides 0
Left Overrides Auto Nav
Right Overrides 0
Right Overrides Auto Nav

NOTICE that blocking a navigation direction will also completely disable that directions overrides.



Block
[✓] Up Blocked
[ ] Down Blocked
[ ] Left Blocked
[ ] Right Blocked
Overrides
Up Overrides 0
Up Overrides Auto Nav

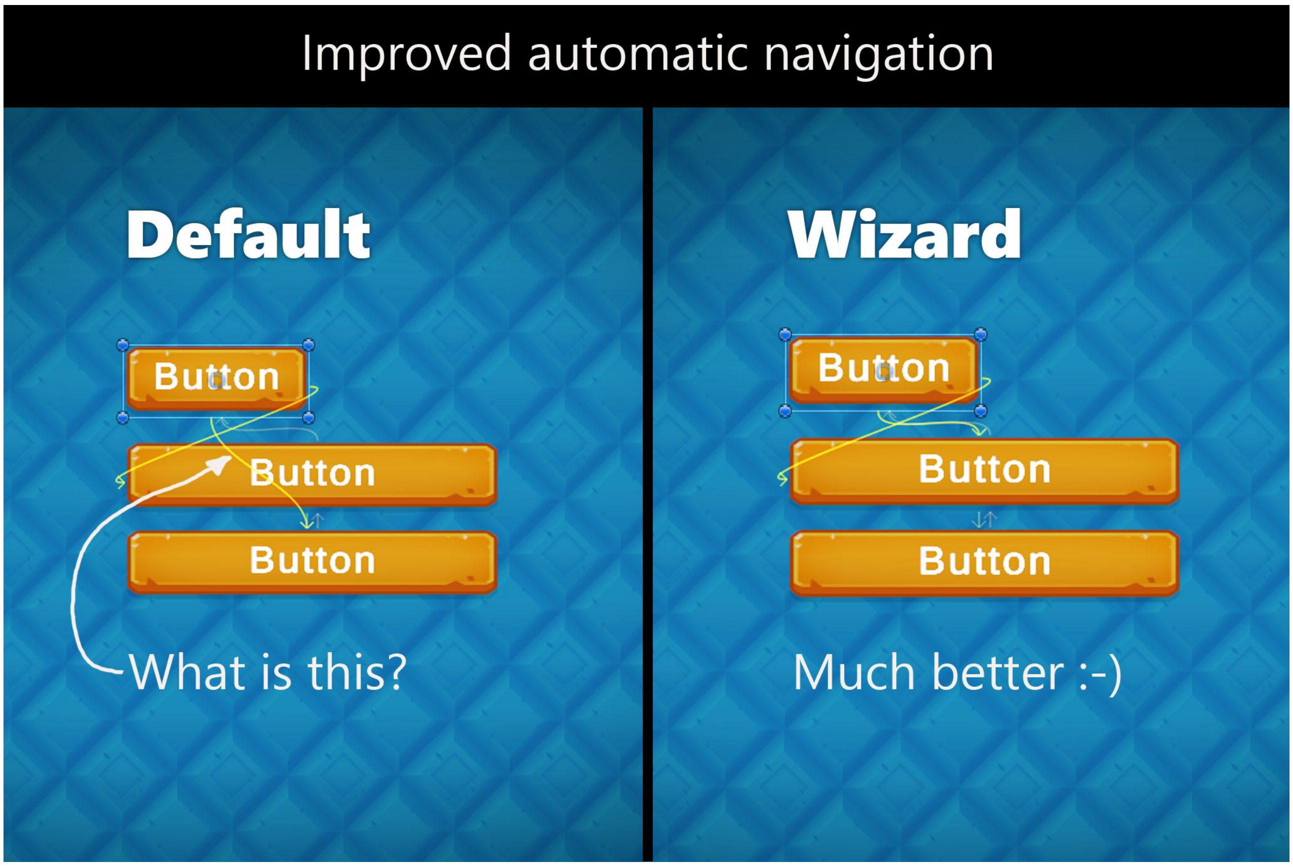
Up/Down/Left/Right Overrides Auto Nav

If enabled then the overrides are not ordered by the list but by the auto navigation (closer ones are prioritized). This is handy for layouts with many elements where a global visual order can not be established easily (grids).

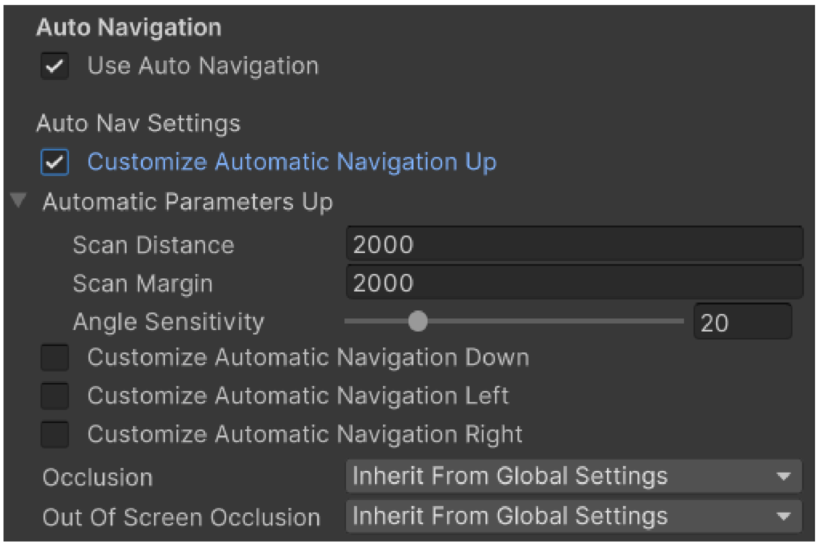
Auto Navigation

What is the „auto navigation“ for?

It is a replacement for Unitys default automatic navigation which sometimes does not get things right. If all goes well you will only occasionally have to configure a wizard by hand.



Below you will find a description of the auto navigation parameters.



Auto Navigation

✓ Use Auto Navigation

Auto Nav Settings

✓ Customize Automatic Navigation Up

▼ Automatic Parameters Up

Scan Distance	2000
Scan Margin	2000
Angle Sensitivity	20

☐ Customize Automatic Navigation Down

☐ Customize Automatic Navigation Left

☐ Customize Automatic Navigation Right

Occlusion	Inherit From Global Settings
Out Of Screen Occlusion	Inherit From Global Settings

Use Auto Navigation

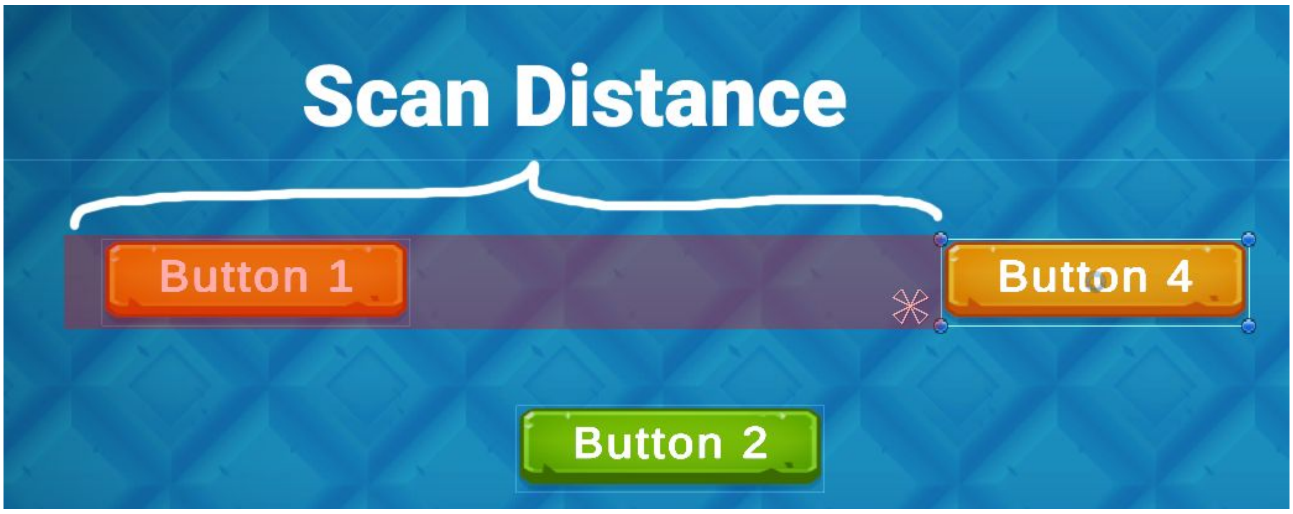
Here you can disable the auto navigation on a per object basis.

Customize Automatic Navigation Up/Down/Left/Right

If enabled the this is the place to configure the auto navigation parameters per object and per direction. The auto navigation default values are coming from the global settings.

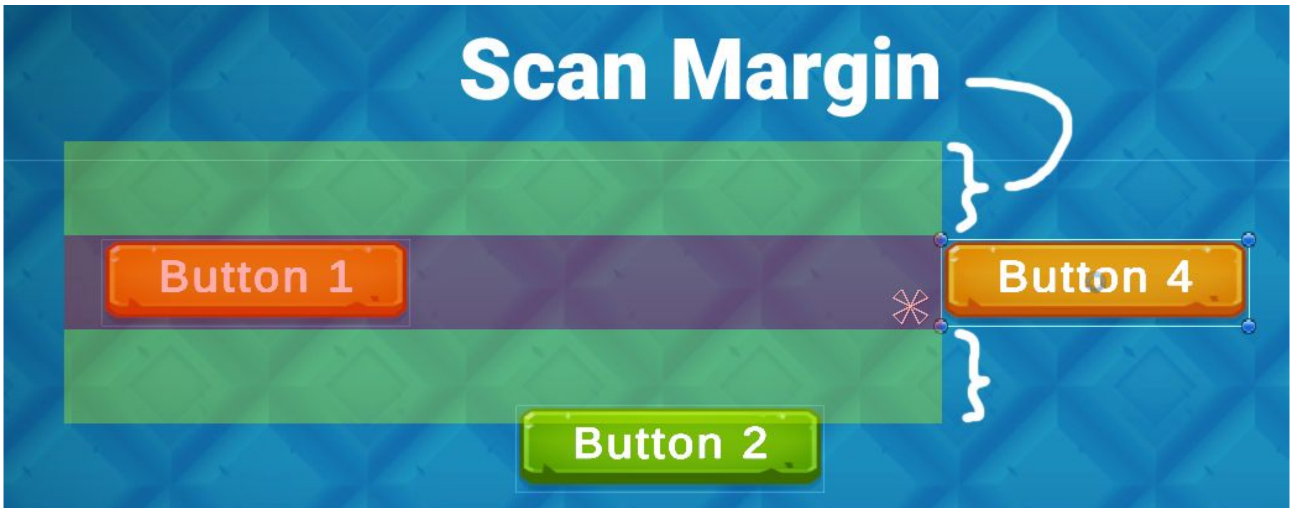
Scan *Distance*

The max distance of the scan area in each direction (red area)



Scan *Margin*

Since growing the selection search area only along the navigation direction would give very limited results the auto navigation also allows for a margin on the sides.

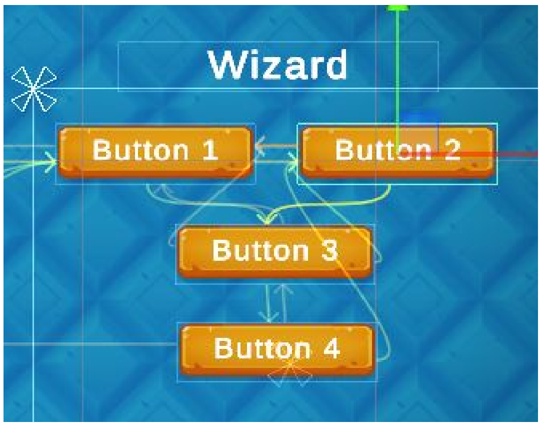


By default both the distance and the margin are huge, effectively taking anything that is to the left of „Button 4" into account for automatic navigation.

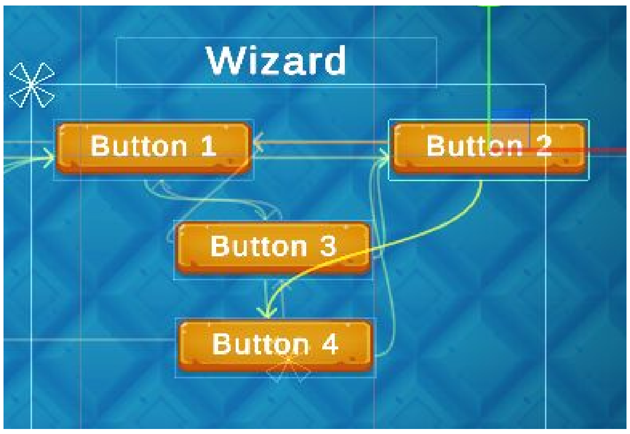
Angular *Sensitivity*

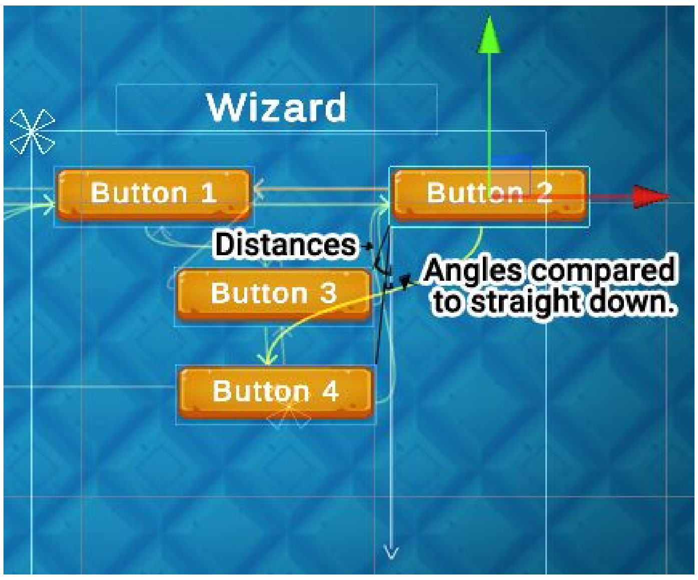
Angular sensitivity defines how to weigh the distance vs the angle. This is used because we want the automatic navigation to take a lot of objects into account (see margin and distance above) yet still keep the navigation direction (angle up/down/left/right) as a priority.

Let's consider this example below (it's part of the UGUINavigationWizardDemo-Native-vs-Wizard demo). As you can see the Button 2 navigates downwards to Button 3. That's fine.



However, if we start moving Button 2 to the right then a some point the downwards navigation target will jump to Button 4.





Why?

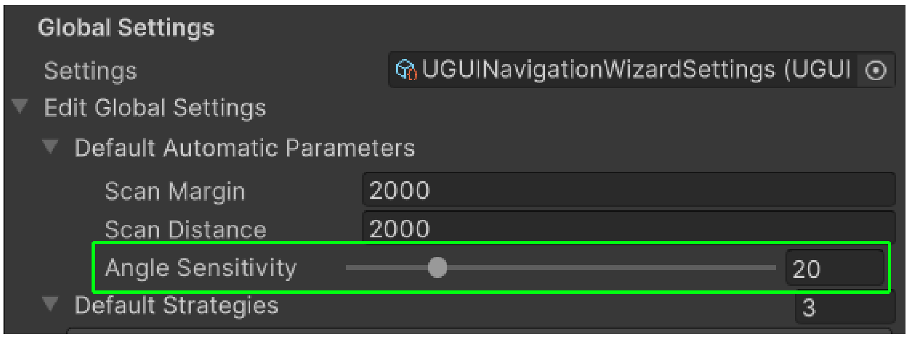
The reason is that the distance of selection targets is weighed by the anglular difference compared to the navigation direction (straight down).

At some point the angle wins over the distance. While „Button 4" is further away from Button 2 it also has a smaller angle difference compared to „straight down". What's more important: small distance or small angle? The angular sensitivity let's you make a gradual decision.

The higher the angular sensitivity the earlier the angle becomes more important than the distance.

Now how to solve our problem here?

Well, by default the „Angular Sensitivity“ is set to 20 in the global settings.



Global Settings

Settings: UGUINavigationWizardSettings (UGUI

Edit Global Settings

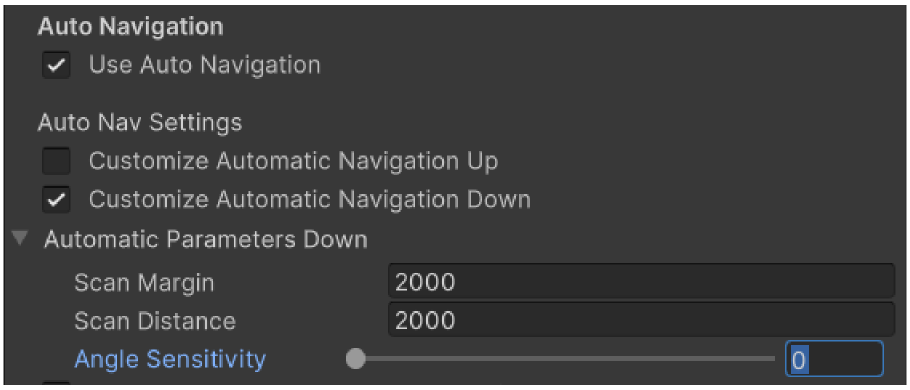
Default Automatic Parameters

Scan Margin	2000
Scan Distance	2000
Angle Sensitivity	20

Default Strategies: 3

We can change this value globally but we can also override it for each individual button direction by enabling the „Customize Automatic Navigation [Up/Down/Left/Right]“ option.

If we set the angle sensitivity to zero then only the distance will be taken into account and we get the result we want without disturbing any other button navigation.



Auto Navigation

[✓] Use Auto Navigation

Auto Nav Settings

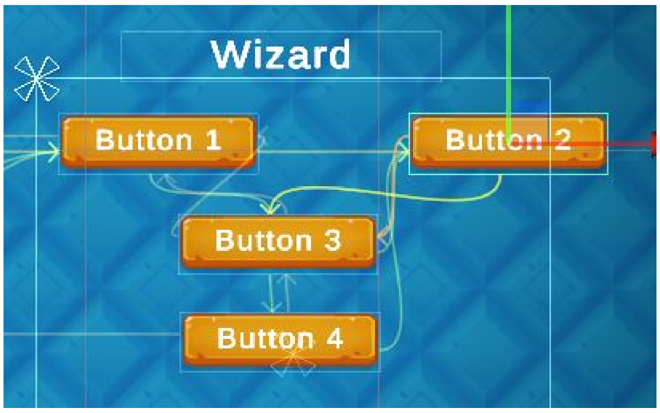
[ ] Customize Automatic Navigation Up

[✓] Customize Automatic Navigation Down

Automatic Parameters Down

Scan Margin	2000
Scan Distance	2000
Angle Sensitivity	0

The result of settings the angle sensitivity to zero for the down direction on Button 2:



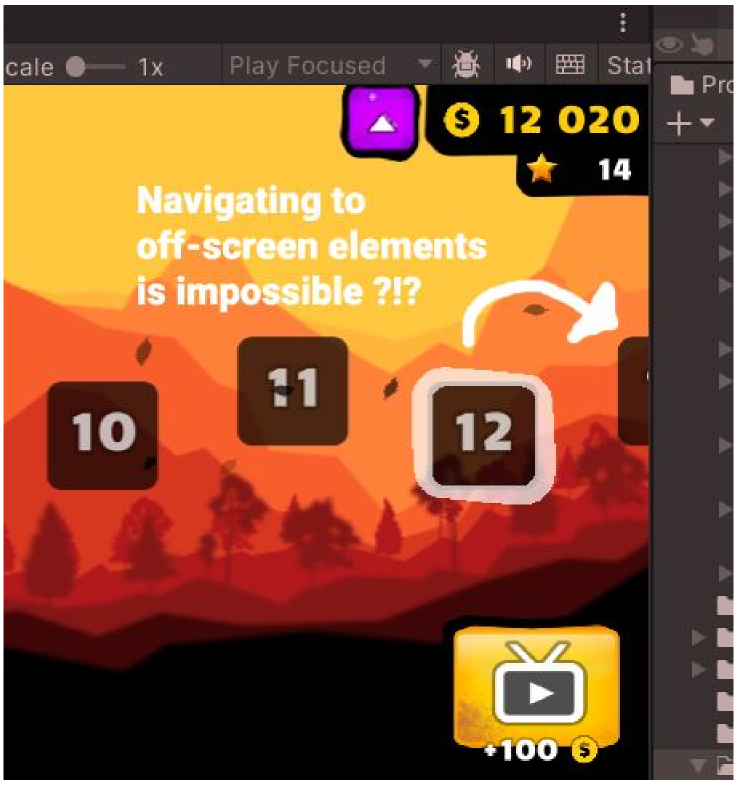
Occlusion

This allows you to control the occlusion on a per object basis.

Out Of Screen Occlusion

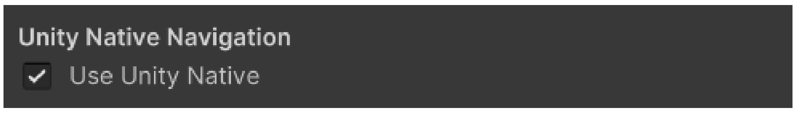
This allows you to control the out-of-screen occlusion on a per object basis.

Example (in this case button 13 is only selectable if „out of screen“ occlusion is disabled):



Unity Native Navigation

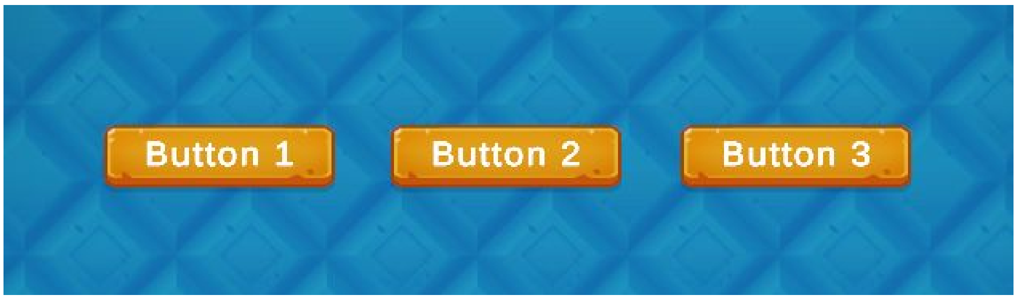
Here you can disable the fallback to using Unity Navitve selection. Not much else to see.

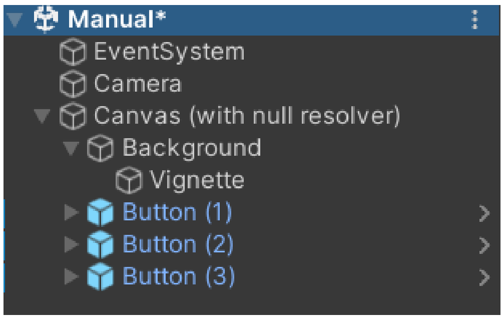

Unity Native Navigation

✓ Use Unity Native

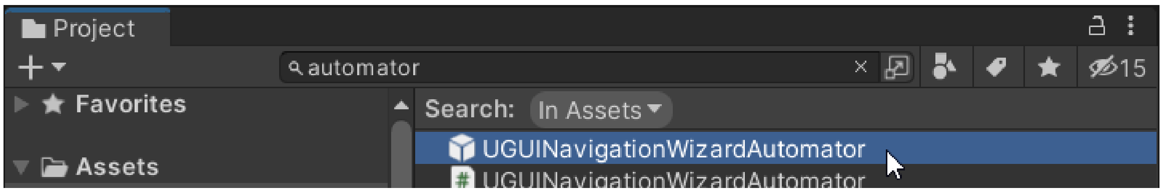
Automator (useful for testing and dynamic UIs)

Let's assume we have this simple scene with a canvas and three buttons:



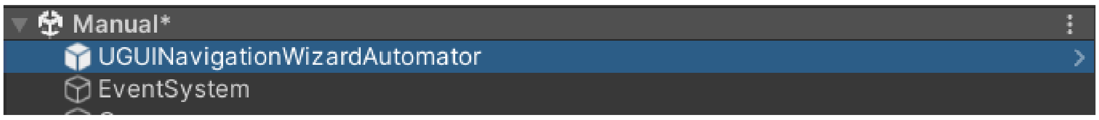


Now if you search in the project view for „automator" you should see the automator prefab.

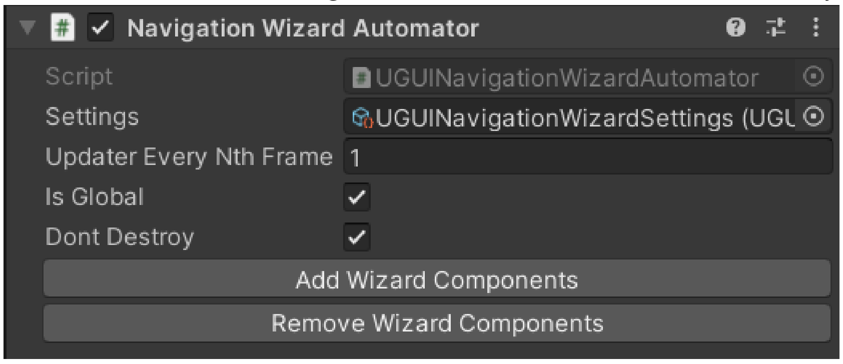


Drag it into your scene.

HINT: While you can use multiple automators I recommend sticking with one for now. You should add this to your main scene (the scene that is loaded at the very start).



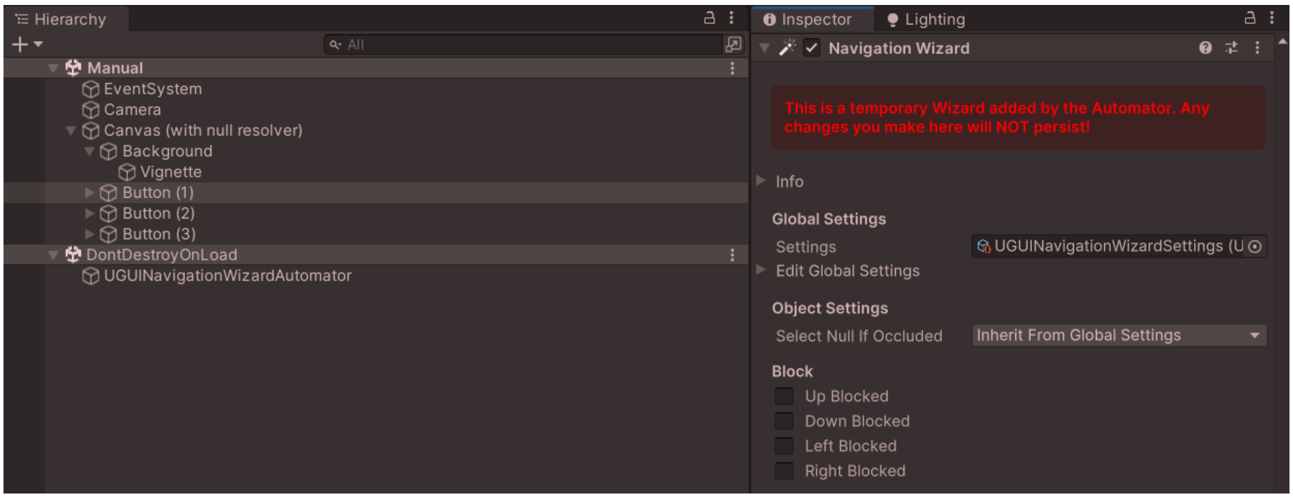
Enable the „Is Global" flag on it (in case it's not checked already).



Navigation Wizard Automator

Script	UGUINavigationWizardAutomator
Settings	UGUINavigationWizardSettings (UGU
Updater Every Nth Frame	1
Is Global	✓
Dont Destroy	✓

And that's it. Once you start your scene a Wizard component should be added to each of your buttons. Please notice the red warning. If you use the automator then the wizard components are only added temporarily (during runtime).



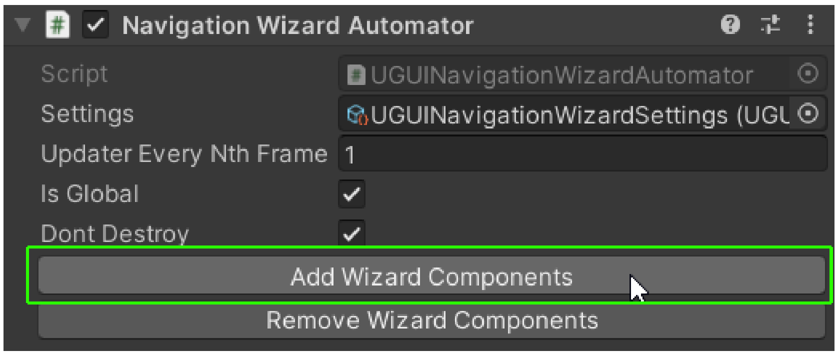
This is a temporary Wizard added by the Automator. Any changes you make here will NOT persist!

Info
Global Settings
Settings: UGUINavigationWizardSettings (U
Edit Global Settings
Object Settings
Select Null If Occluded: Inherit From Global Settings
Block
Up Blocked
Down Blocked
Left Blocked
Right Blocked

At runtime you can play around with the local parameters of the wizard.

NOTICE: If you change the global settings then these changes will persist, even if the component has been added by the automator.

If you want to add the component permanently then you can use the „Add Wizards" button during edit mode:



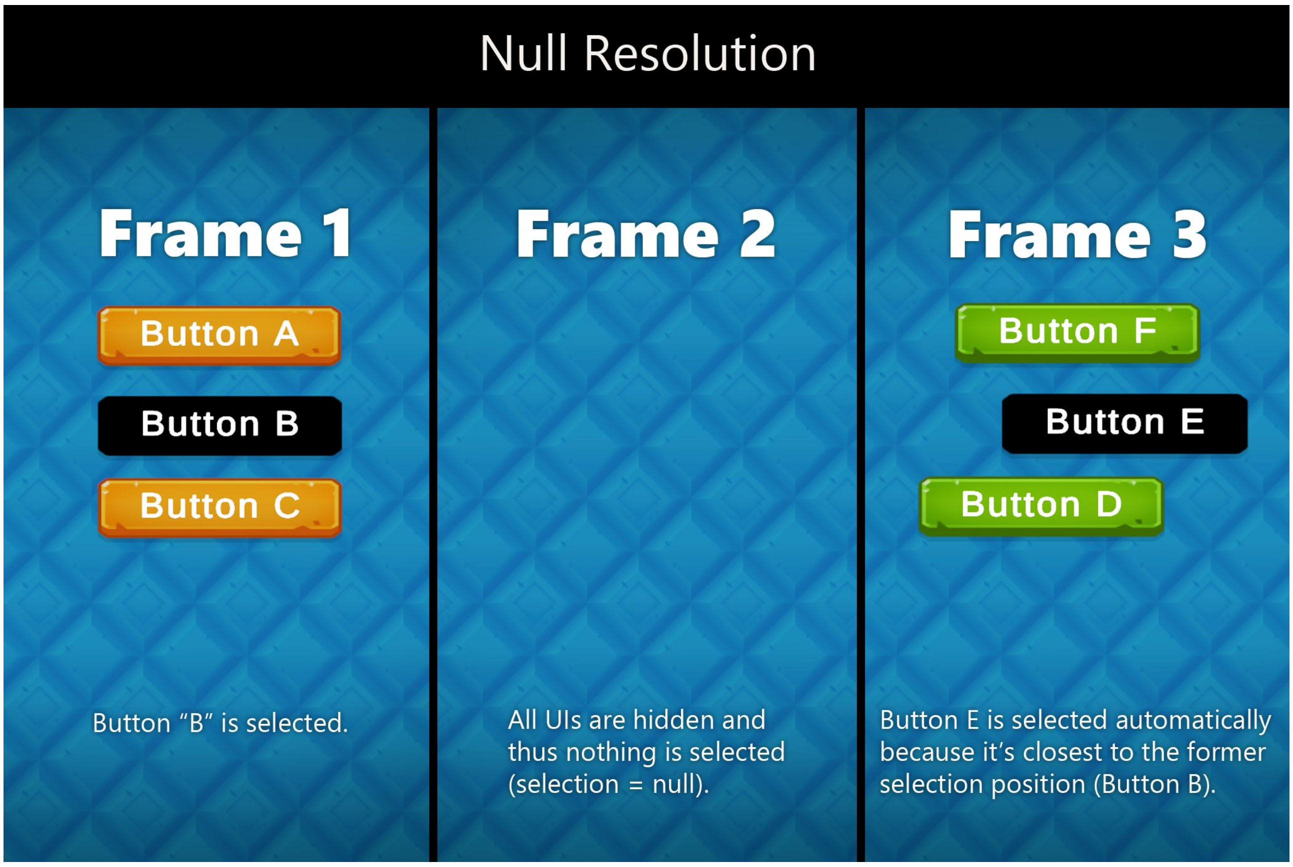
Script: UGUINavigationWizardAutomator
Settings: UGUINavigationWizardSettings (UGL
Updater Every Nth Frame: 1
Is Global: ✓
Dont Destroy: ✓

HINT: If you disable the IsGlobal flag on the automator then it will only take the selectables into account that are children of the automator transform. This is useful for testing scenes independently.

NOTICE: You should not have multiple global automators.

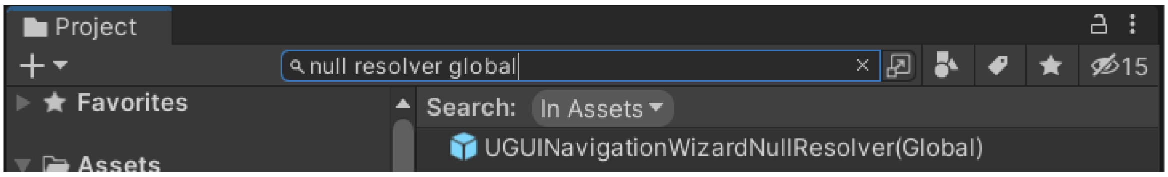
Null Resolution

What is null resolution and why should I care?


Null Resolution
Frame 1
Button A
Button B
Button C
Button "B" is selected.
Frame 2
All UIs are hidden and thus nothing is selected (selection = null).
Frame 3
Button F
Button E
Button D
Button E is selected automatically because it's closest to the former selection position (Button B).

For users with a mouse or on touch null resolution is not a big deal, but for console or keyboard-only players it is vital because if the selection becomes null then they can not longer trigger any buttons.

By default null selection resolution is not enabled. To use it you have to add the „Null Resolver" prefab to your scene.



null resolver global

Search: In Assets

UGUINavigationWizardNullResolver(Global)

HINT: While you can use multiple null resolvers I recommend sticking with one for now. You should add this to your main scene (the scene that is loaded at the very start).

The null resolver will check every frame (Update()) if anything is selected and if not then it will select the selectable that is closest to the last known screen position.
If a valid selection exists it will remember the screen position of that selection for future reference. That way the null resolution will always start from the most recent position which ensures a nice continuity.

HINT: The NullResolver has some static a SelectNull() and a SetAllPaused(bool pause) methods. You can use these to halt null resolution. It also has a Resolve(..) method which you can use to trigger the resolution yourself.

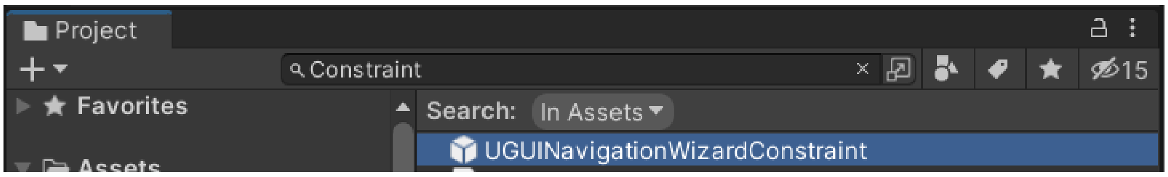
HINT: If you disable the IsGlobal flag on the resolver then it will only take the selectables into account that are children of the resolvers transform. This is useful for testing scenes independently.

NOTICE: You should not have multiple global null resolvers.

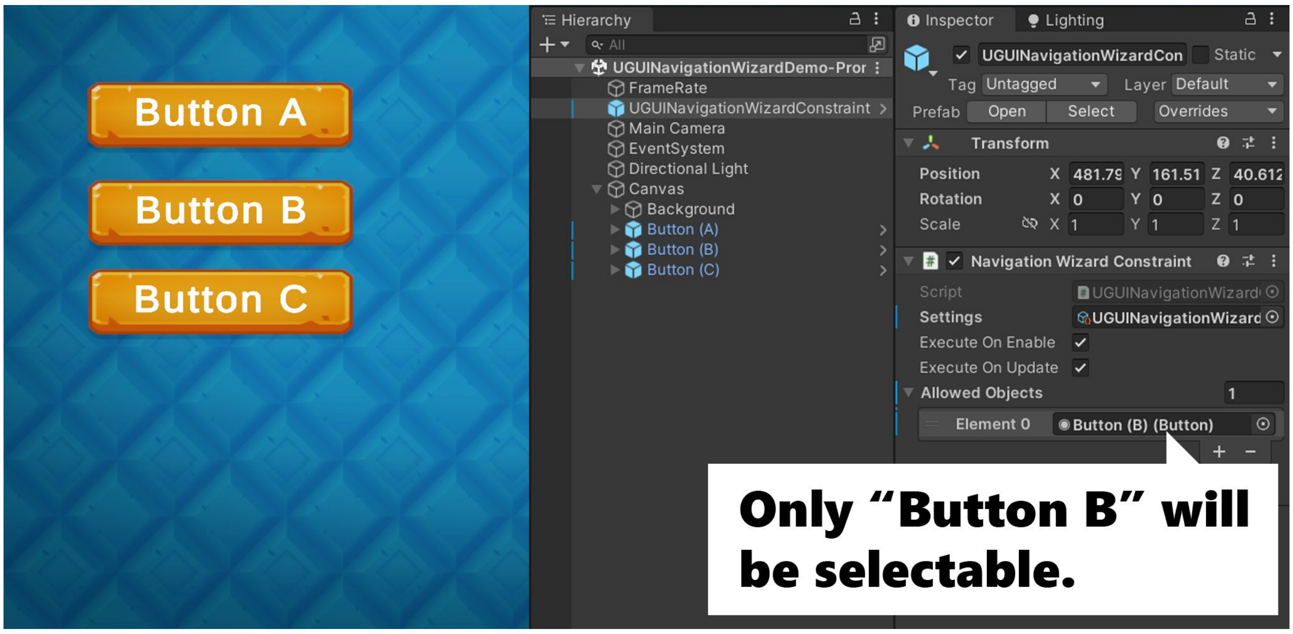
Constraints

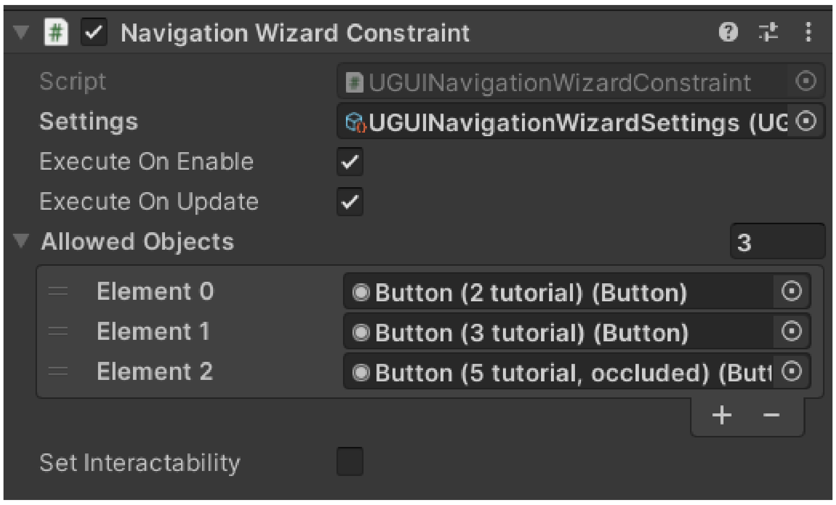
Constraints are basically a list of selectables and the system will ensure that ONLY these are selectable. This is very handy for making tutorials.

There is a constraint prefab but you can also just add the constraint component to your game object.



HINT: You can have as many constraints active at the same time as you want but I would recommend to only ever use one at the same time (or at least just have only one with „Execute On Update" enabled). The lists of objects are accumulative. Each activated constraint will add to a global list (stored in the settings as a non-serializable list).





Navigation Wizard Constraint

Script	UGUINavigationWizardConstraint
Settings	UGUINavigationWizardSettings (UG
Execute On Enable	✓
Execute On Update	✓
Allowed Objects	3
Element 0	Button (2 tutorial) (Button)
Element 1	Button (3 tutorial) (Button)
Element 2	Button (5 tutorial, occluded) (Butt
Set Interactability	

Execute On Enable

If the constraint object is enabled then it will perform the changes to the global list of allowed objects.

Execute On Update

If enabled then it will perform the changes to the global list of allowed objects in Update() (only does so if changes are needed). This is useful to support UI that enable/disable buttons at runtime.

Allowed Objects

The objects that should be added to / removed from the global list of allowed objects. Only these objects will be selectable.

Set Interactability

If enabled then selectables that are not in the „AllowedObjects“ list will have their „interactable“ property set to false and restored to true once the constraint is disabled.

Frequently Asked Questions

The navigation visualization is not always accurate in Scene View.

Please make sure the Game View is focused at runtime. If you focus the scene view at runtime then the navigation will take the scene view camera into account and thus not display the game view visualization for the game view but the scene view (which my be confusing).

SOLUTION: In doubt always test with the game view focused.

My images are not selectable even though they are clickable, why?

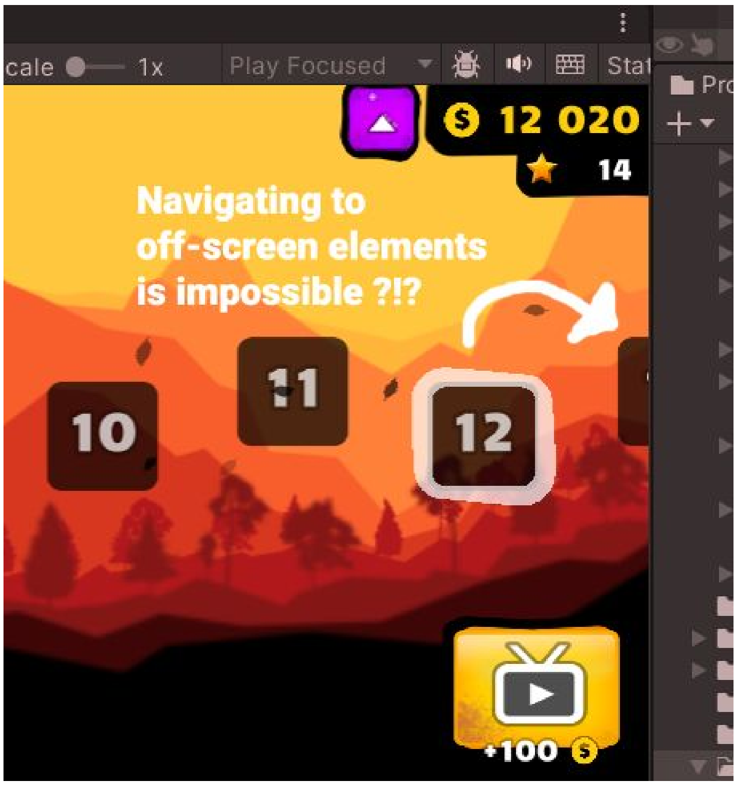
In order for the selectable system to recognize a UI element as being selectable you have to add a Selectable component. Buttons and inputs are derived from the Selectable class so they don't require you to add one.

However, if you have an image and simply add an EventTrigger component to make it clickable then that does not make it a selectable right away, even if it can be clicked.

SOLUTION: Please add a Selectable component to your images.

My scroll view buttons are not selectable if off screen, why?!?

That's because by default all off-screen elements are considered to be occluded.



Solution: Disable the out-of-screen occlusion on the scroll view buttons.

Where is the cut-off point for occlusion?

It’s the center point on the rect transform. NOTICE: That’s not the pivot, but the actual rect.