No Touch GUI for VR™

Allow users to interact with your VR projects without the need for an external controller

THANK YOU for purchasing!

It's a huge deal to me. So thanks!

The No Touch GUI for VR system was developed by Sean Smith *for*

B2CON NECK GAMES

ABOUT "NO TOUCH GUI FOR VR"

How It Works:

The crosshair does a raycast in the direction you're looking, for a specified amount of distance which can be set and modified, and detects any object with the BNG_ZapperAction script attached, so anything you put in the crosshairs, your users can interact with. This system sets up a series of visual cues for interaction using bar and radial timers. You can customize the timer's color, its activate/deactivate and cool down times, and call functions on game objects during activation and deactivation phases using the SendMessage function. I tried to make it easy to use so you can just drop the prefabs on your object and it works with minimal setup. That being said there are some things you need to consider when preparing to use this in your project, which we will discuss on the next page.

Why I Made No Touch GUI for VR:

I developed this system for VR games that require a controller so those people who don't have a controller can still get around and interact with your project. It's not great for "high twitch" gaming but would work nicely for casual games were the player works at their own pace and it would be great for targeting objects for turret/shooter games. I have included some uses for these in the demo scenes.

Development:

This system spawned from my need for an autowalk toggle, similar to that in the Dive Tuscany Demo. It is heavily sprite based for several reasons, abillity to use sorting layers, lightweight, easier to manipulate and create (for me), etc... But, I am working on integrating it with the new uGUI and adding in other options besides timer bars to indicate wait times, expanding panels with multiple options, upgraded menu styles and other cool features for a total "no touch" gui experience. This product represents a lot of work and I hope to make this the definitive VR GUI asset on the Unity Asset Store, so I will work hard and strive to improve it.

Publishing from the Demo Scene:

I had a lot of trouble until I figured this out, so I will tell you now and save you the headaches. If you are using this for a VR project, I highly suggest you keep separate projects for separate systems. If you're using the Dive Plugin in your project, you can only publish to Android or iOS (which means if you want to publish to desktop or web, you have to remove the Dive folder and it's related plugins). If you have Dive and Cardboard in your project and you try to publish to Android with the VR Player for Durovis Dive, it will not work (not sure about iOS). You can publish to Android with the VR Player for Google Cardboard (with or without Dive in your project). You also can't seem to do much if you have Oculus in there. I don't have one so I can't test it, but I hear you can't have Dive in your project either. All those VR things share common java libs I think and it causes a jam up. ***So if you are doing a VR project, CHOOSE ONE VR SDK AND STICK WITH IT.***

BEFORE YOU START...

Since I can't include external assets in my project, there are a few things you need to do to get the Demo scene working properly whether you're using it for VR or regular 3D. ***The prefabs in the project will have missing components and not work until these steps are completed!***

1. Download the Durovis Dive Unity Plugin from https://www.durovis.com/sdk.html and the Google Cardboard Unity Plugin located at https://developers.google.com/cardboard/unity/download

2. Open your project and import the No Touch GUI for VR Asset

3. Import the Durovis Dive & Google Cardboard packages by going to *Assets -> Import Package -> Custom Package* ***and select the unity package when prompted. On the import dialogue make sure you select*** *Import All.*

4. If it's not already there, import the Standard Character Controllers by going to *Assets -> Import Package -> Character Controller.* ***On the import dialogue make sure you select*** *Import All.*

5. Navigate to the "NoTouchGUI Assets/Scripts/external scripts" folder and drag the file called "DiveFPSControllerForNotouchGUI.js" into the "Dive" folder.

6. Navigate to the "NoTouchGUI Assets/Scripts/external scripts" folder and drag the file called "FPSInputControllerForNoTouchGUI.js" into the "Standard Assets/Character Controllers/Sources/Scripts" folder.

7. Navigate to the "NoTouchGUI Assets/Prefabs" folder and select the "VR Player for Durovis Dive" prefab, it will have a missing script reference. You need to navigate to the "Dive" folder in your project view and drag the "DiveFPSControllerForNotouchGUI.js" script into the missing script slot in the inspector. You should do this for the "Player for VR" prefab as well but it has been depreciated.

8. Navigate to the "NoTouchGUI Assets/Prefabs" folder and select the "NON VR PLAYER" prefab, it will have a missing script reference. Navigate to the "Standard Assets/Character Controllers/Sources/Scripts" folder in your project view and drag the "FPSInputControllerForNoTouchGUI.js" script into the missing script slot in the inspector. Do this for the "Vr Player for Google Cardboard" prefab as well.

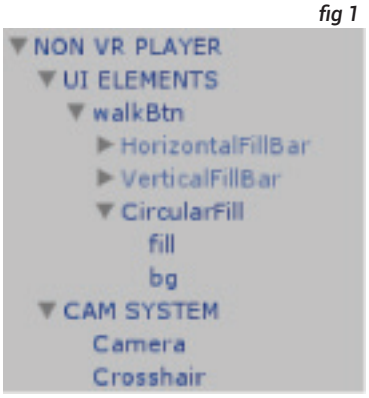
9. Now, you can open the Demo Scene found in the "NoTouchGUI Assets/Scenes" folder and activate either the "Player for VR", "VR Player for Durovis Dive", "VR Player for Google Cardboard" or the "NON VR PLAYER" game objects and be able to move about the scene. You can use the keyboard (WASD) keys to move, or look down and use the autowalk toggle.

GENERAL USE *(Using Prefabs)*

Using the prefabs is the easiest way to get up and running since the code is all set to go.

Step 1: Setting up the Player

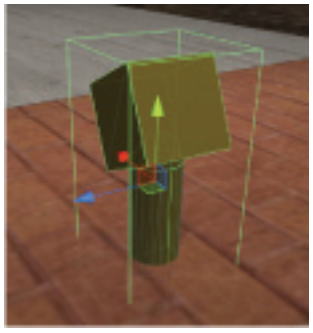
fig 1



Just drag any Player prefab into the scene and you are ready to start creating buttons. These prefabs already have an autowalk toggle built in. Feel free to adjust the settings under the ***BNG_Zapper*** *and* ***BNG_ZapperAction*** *scripts & their* ***Sprite Renderers*** *to customize colors for your crosshairs and the length of the ray it is casting. You can even set the colors to transparent for an immersive first-person VR experience.*

***Note:** The FPS Controller has been customized for this in order for the camera and crosshairs to move together. The camera and crosshairs need to be in a group together (fig 1), and the FPS Controller needs to have a reference to this group. The prefab has the code you need when setting it up from scratch.*

fig 2



Step 2: Adding A Button

To create a button, you can start by dragging the INTERACTION PODIUM prefab into the scene. This is just a cube and a cylinder with a yellow texture applied, it is what you see in the demo and makes for a nice button holder (fig 2). Next we will setup the custom button by placing one of the ICON SPRITES from the Sprites folder into the INTERACTION PODIUM game object. You may have to reset the position transform on the sprite to 0, 0, 0 and then move it into place on the face of the podium (fig 3). The podium and it's components' layers are set to "Ignore Raycast". This is because those objects have physics colliders as well, and our raycast will hit these and think we're not on our button. Setting their layer to Ignore Raycast will prevent unwanted results from the script, just make sure your button is not set to Ignore Raycast! Now comes the fun part :)

fig 3

Select the game object you want to interact with and click "Add Component", select Scripts -> ***BNG_ZapperAction****. This is where the magic happens. Your button will now function, although we still have some setting up to do, before it will do anything for us.*

GENERAL USE (cont'd)

Step 3: Configuring the Zapper Action Script

*This script is the **core of the No Touch GUI System** and it has a lot of options you can set to create different types of buttons, toggles, menu items and even enemies. Let's take a look at the parameters:*

[depreciated] ***Using Sprites:*** *If you are using this on a mesh object you'd want this off as a lot of the options are sprite based and therefore are looking for a Sprite Renderer component, for mesh objects, these options will not work.*

Change Color on Hover: *This will change the color of the object with this script attached, this is good for indicating the object is being activated and works best when the sprite is a white base, so the color is easier to see.*

Change Color on Activate: *This will change the color of the object once it's activated.*

New Color on Hover: *This is the color that the button will change to if the Change Color options above are checked.*

Rotate Object with Camera: *This will rotate the entire button object & its children so they're always facing the camera.*

[depreciated] ***Tooltip On Hover:*** *If you would like to display a tooltip, check this box. If you have a tooltip in Tooltip Game Object but this is unselected, it will assume you want to use a tooltip.*

Rotate Tooltip and Timer on Hover: *If seleced, this will rotate only the tooltip & timer objects but not the button sprite, to face the camera as seen in the demo with the gun & ammo clips. Rotation looks best if the tooltip is nested in the timer for this.*

Tooltip Game Object: *This is the game object (typically a sprite) that will pop up when you hover over the button object. Leave it blank if you do not want a tooltip to display.*

[depreciated] ***Require Timer:*** *If you want the object to display a timer indicator, check this box, if you check this and do not have a timer inside your button game object, it will turn itself off.*

Replacement Sprite: *Once activated, the button can change its sprite to display a different sprite (ex: demo door switch).*

One Time Use: *If selected, the button will only activate once and make it so it cannot be activated or deactivated again. This is done by changing its tag to "Untagged". You can also script this easily, for any game object you don't want to be active.*

Reset on Activate: *If selected, the button will only trigger the activation function and not the deactivation function. It will still use the wait time parameter inbetween activations. Useful for menu items and other single state buttons.*

Time to Activate: *This is the amout of time (float) it will take to activate the item.*

Time to Deactivate: *This is the amount of time (float) it will take to deactivate the item.*

Wait Time: *This is the amount of time (float) between activation and deactivation periods, also known as a cool down period.*

Call Function On Activate: *Since the system uses the Send Message function which takes a string, this is the name of the function you want to call when the button is activated.*

Game Object to Activate: *A reference to the game object that contains the function you are trying to activate. This is a game object that contains a script with a function named whatever you enter for "Call Function on Activate".*

Call Function on Deactivate: *This is the name of the function you want to call when the button is deactivated.*

Game Object to Deactivate: *Reference to the game object that contains the function you are trying to call on deactivate. This is a game object that contains a script with a function named whatever you enter for "Call Function on Deactivate".* ***Note:*** *If you leave this blank and have a string in the "Call Function on Deactivate", it will use the game object you selected for "Game Object to Activate".*

GENERAL USE *(cont'd)*

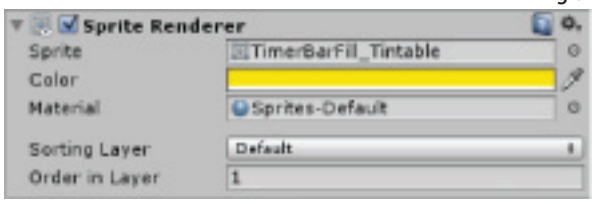
Step 3.2: Adding a Timer Prefab

fig 4

Once you have your Zapper Action script configured to your liking we'll need to setup the Timer Fill Object to indicate that the object is being activated (if you're using them, that is, I'll assume you are since you're so cool) . You can do this part before, while or after you've configured your Zapper Action script.

First, find the Timer prefab you want to use (HorizontalFillBar, VerticalFillBar or CircularFill) in the prefabs folder and drag it into the button you attached the ***BNG_ZapperAction*** *script to (so the timer is a child of the button object) and position it where you like (fig 4). The VerticalFillBar option is really just the HorizontalFillBar prefab rotated 90 degrees. You can change the color of the bars as you see fit by changing the color on the "fill" and/or "bg" objects' Sprite Renderer (fig 5). I am working on writing a shader for the radial timer to be tintable as well. If you need the PSD file to create colored radial timer sprites or materials, let me know and I will send it to you.*

fig 5



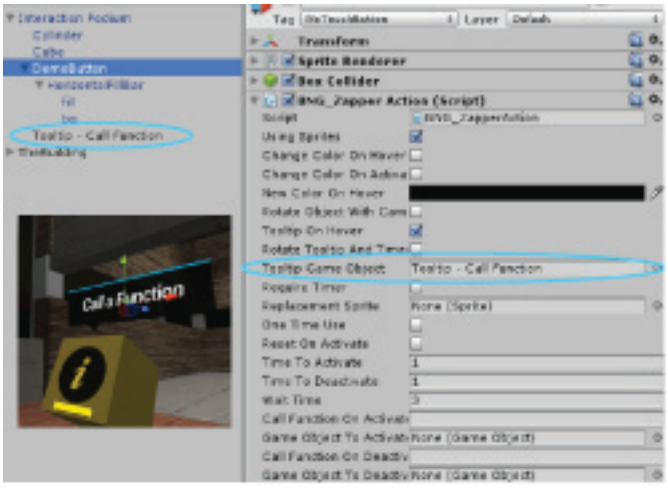
Whatever you choose to use, there are a couple things you should note:

- The Timer Fill Objects need to keep their name as they are in the prefabs so the system will work. I am working on updating all timer bars so you don't have to worry about their name.

- You do not need the "bg" game object in the timer, it can be deleted, but the "fill" game object must remain a child of the timer prefab and keep its name as "fill".

Step 3.3: Adding a Tooltip to the Button

fig 6



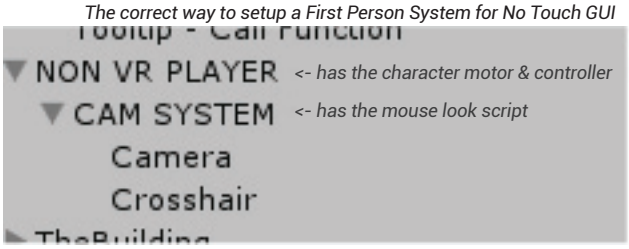
Adding a tooltip is as simple as creating the game object you want to use as the tooltip, dragging it into position and then linking it in the Zapper Action script (fig 6). In the demo, we simply made our tooltip images as .png files, and dropped them into our project. The Zapper Action script will hide them when you run the scene and show them when you the button is hovered over.

CUSTOM FPS CAMERAS

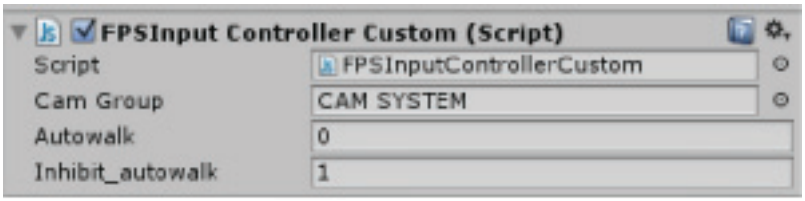
USING CUSTOM PLAYER CONTROLLERS

The main problem you run into when using other player controllers than the one in the demo is that the crosshair sprite requires a certain hierarchy to work. It needs to be inside a game object (ex: Cam Group) and on the same level as the camera. It also needs to be within the boundaries of the character controller's capsule collider, so it doesn't clip when you hit a wall.

The correct way to setup a First Person System for No Touch GUI



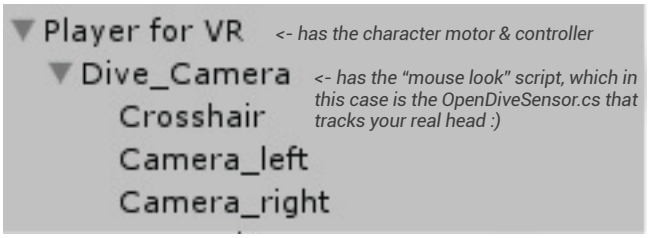
Then you need to adjust the crosshairs so it is right in front of the camera and scale it down so you can see it in the game mode view. I setup the crosshairs as a sprite because I like that you can control the Sorting Layers, so I create a layer above all others, call it UI and set the sorting layer to 9999. You should also adjust the near clipping plane of the camera to 0.01. Once the crosshair is in the proper place, you need to get it to move with the camera. This is why you put it in an empty game object, for example say it's called "Cam Group" and it contains the crosshairs and your camera, the Cam Group would only have the Mouse Look scripts attached (remember to set axis on Mouse Look to X ***and*** *Y), as it is your characters "head". The Player, in which the Cam Group is a child of, would have the charcter motor, FPS Controller and any other scripts that deal with moving the whole group of objects, or the player's whole "body". The point is, when you set it up this way, the Unity Standard FPS Character Controller will move properly, but it will not move in the direction of the camera because it would normally have the mouse look scripts there so it doesn't know which way you are facing. You need to add a reference inside the FPS Input Controller to connect the Cam Group that has your mouse look script, to the FPS Controller. If you open the Unity Standard Asset First Person Controller and find FPSInputController.js, you can replace this line of code: "****motor.inputMoveDirection = transform.rotation * directionVector;****" with this: "****motor.inputMoveDirection = camGroup.transform.rotation * directionVector;****" where* ***camGroup*** *is a reference to the object that holds your camera and crosshairs (with the mouse look script on it). That will allow your character to move forward in relation to where you are looking.*



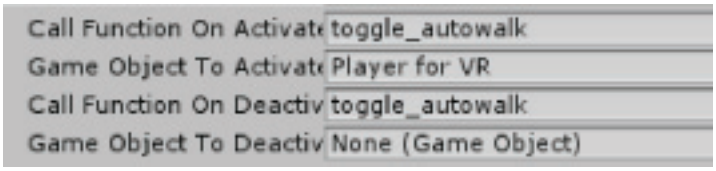
The modified version of the FPSInputController Script with the autowalk feature added in from the Durovis Dive plugin. I added this for the demo so you could see it on the webplayer, but this really wouldn't be needed unless you were in a VR setting. But hey, who am I to judge...

DUROVIS DIVE for VR

SETTING UP DUROVIS DIVE WITH NO TOUCH GUI



Similar to the above, the same problems arise when adding a crosshair to your base Dive_Camera group. It's actually easier than the standard FPS Controller because the Dive Prefab was made to use the Dive_Camera group as a head so you don't need to pass reference to it, just setup the crosshairs like the standard asset on the last page. Dive comes with a FPS Controller just for it, so you would attach this to the "Player" group and inside that group would be your Dive Cam group with the OpenDiveSensor.cs that is the same as your mouse look. The Dive FPS Controller needs a reference to the Dive_Camera, which is also a prefab with the Dive package, so don't forget to connect them. When testing, you can use your keyboard to move about, but things are much different when the headset is on :)



To connect it to your No Touch GUI button, you can call the built in function "toggle_autowalk" in the Dive FPS Controller script. Just put "toggle_autowalk" as the "Call Function on Activate" and "Call Function on Deactivate" parameter and drag the game object with that script to the Game Object to Activate field and you should have a working autowalk feature. The Player for VR prefab that comes with this package includes a working model for the Dive FPS character controller.

If you're stuck and need any help, let me know!

No Touch GUI and UFPS

SETTING UP NO TOUCH GUI WITH UFPS

Thanks to the help of @DJNombe, we have a somewhat workable prototype for using Durovis Dive + No Touch GUI with the UFPS system. It should work just fine with the regular UFPS package though. I will test this out more in the future and give a detailed report about using No Touch GUI for VR with a UFPS + VR Package. In the meantime, you can check out this thread for more info: http://t.co/kJENyux8Z9

TROUBLESHOOTING

HELP! My Button Is Not Working!

Never fear! I came across this many times in development and have a few tricks you can try. This can be caused by numerous things. Here are some things to check:

- *Make sure the function you are trying to call is in a script of the Game Object to (De)Activate*
- *Make sure it has the Zapper Action script attached*
- *If there are other objects with colliders touching the button's collider, set their layer to "Ignore Raycast"*
- *Make sure the button isn't set to "Activate Once" or it will only work once*
- *Make sure your crosshair object has the Zapper Script attached*
- *Try increasing the ray length variable on the Zapper Script, or the size of the physics collider*
- *Make sure the button's layer isn't set to Ignore Raycast*
- *Check the activation/deactivation & cool down times to see if you are just waiting for it to finish*
- *Check the output in the console, sometimes a helpful message is waiting for you*
- *Did you apply any settings to the Zapper Action script that would indicate a change in state?*

My Button acts like I am still hovering over it when I'm not...

This issue (which has been updated in v.1.10.3) can be caused when there is nothing around your button for the Raycast to collide with, such as the sky. The raycast won't unload because it still thinks the last thing you collided with, is still colliding with it. Update the BNG_Zapper script if you want to avoid this error.

My Crosshair doesn't seem aligned to the objects I'm trying to hit...

This issue occurs in VR, since we're just using a sprite for the crosshairs and not an actual Unity GUI item, when there are two camera's present the perspective make its look like the crosshair is fine, but when you try to activate objects, you may have to point to the side or above/below its collider to trigger the scripts. The only thing I can say about this is just experiment and move the crosshairs around, closer/further from the camera and/or up and down a little too. Usually this is caused by the z axis distance from the cameras.

Well, for now those are the major issues I can think of but if you have ANY other issues with this system, please contact me first and I will not hesitate to help you out! *You can contact us on our website, www.baconneckgames.com/contact-us, on twitter @BaconNeckGames #NoTouchGUI or through email at dev@baconneckgames.com. I am happy to help you get the most out of this system!*

GETTING STARTED *(Basic Setup - No Prefabs)*

In order for the No Touch GUI system to do it's job there are a few things you need to setup. I have included a lot of prefabs in the demo scene to help you get up and running as quick as possible, if you're not using the prefabs, here's what you need to know:

***1**. You will need to create the proper hierarchy of Player -> Camera Group -> Camera to put your crosshairs in. You do not need a crosshair, the script will work on any object. In a standard setup, the "Player" will have the movement scripts and the "CameraGroup" will have the mouse look scripts. The point is that the crosshairs need to move with the camera, so setting it up like this works to solve this problem. In the demo I use a simple sprite for the crosshairs, but you have to make sure it won't get clipped by the camera. To solve this, I put it really close to the camera, lower the camera's near clipping plane to 0.01 and shrink the scale of the sprite so it looks normal. Make sure the crosshairs are within the character controller's capsule collider, that way when you get up on a wall the crosshairs don't disappear behind it. Check the Player prefabs in the demo scene if you need help.*

***2**. When your crosshairs are in place, you can add the **BNG_Zapper** script to it and adjust it's settings. This is the script that sends out a raycast and allows the interaction to happen.*

3**. Then just attach the **BNG_ZapperAction** script to any game object you want to interact with and start customizing it's properties to get the behavior you want. **Each property has a tooltip in the inspector that explains what it does that will activate when you hover over it.

***5**. If you choose to use a timer indicator, there are a few prefabs you can choose from to drop into your object (so the timer is a child of that object). The name of the timer SHOULD NOT be changed for the system to recognize it. You can have multiple timers just as long as one is active at a time.*

6**. In order for you to be able to manipulate objects, you need to create a simple script with a function that has the same name as the string you use in the "Call Function On Activate" field. Then you attach that script to the object you want to manipulate (or any object really), and drag that object into the "Game Object to Activate" field. **NOTE: If you leave the "Game Object to Deactivate" blank, it will use the same game object as the "Game Object to Activate".

Take Advantage of the Demo Scene!

I spent a lot of time on the demo scene getting everything to work nicely together. There is a lot of good code there that you can reuse, a lot of example uses, as well as some good prefabs to get you started. Use those whenever you can to avoid setup issues, but I will explain how to create these items from the ground up later on.