Simple Animated 3D Buttons - Deluxe Edition

Intended usage

The buttons are create to simulate the same setup as the built in Unity Canvas buttons; add them to the project, set up the visual properties (Button 3D mesh, position, rotation, and scale), and finally set the methods and when to trigger them.

Initial Setup

1. **Drag "ButtonPrefab" into your scene where you want the button to be placed at.**

2. **Drag "ButtonRaycastControllerGO" into your Scene/Scene Hierarchy** (or drag the script "ButtonRaycastController" from the CoreLogic folder inside the root Simple Animated 3D Button folder to an active gameObject.

The Prefabs

- ButtonPrefab

This prefab contains only a base mesh and a button mesh, with no sprites or 3Dtext gameobjects attached to it.

- ButtonPrefab3DText

This prefab contains everything the ButtonPrefab does, whilst also including a 3D Text gameobject, that allows the user to easily swap out the provided sample text to reflect the intended function of the button.
Changing the text can be done through selecting the 3DText gameObject and using the inspector to change the text. Changing the font will require you to import the font, and then drag the "Font Texture" that gets created upon importing custom fonts, and applying this font to the DepthTextMaterial, due to the custom shader that is used to prevent the 3D text from rendering through other objects.

- ButtonPrefabSprite

This prefab contains everything the ButtonPrefab does, whilst also including a sprite 3D Text gameobject. This allows the user to replace the provided sprite with something else, to populate the entire button with custom graphics.

Customizing the 3D Buttons

Wanting to change the buttons out to ones that are more stylistically appropriate to your own project is expected and encouraged. This can be done in two different and simple ways.

Changing Mesh - Child Mesh Edition

(Pros: Less likely to produce unexpected results, Cons: Scene Hierarchy is less minimalistic)

1. Drag and drop your custom 3D button mesh onto the ButtonMesh Child object of the ButtonPrefab, making your own mesh a child of the ButtonMesh.
2. Position your own mesh to align with the height of the original ButtonMesh
3. Select the ButtonMesh Child object of the ButtonPrefab, and use the inspector to turn off the Mesh Renderer component to hide it.
4. Drag and drop your custom 3D base mesh onto the ButtonPrefab object, and position and scale it accordingly.
5. The BaseMesh object can be deleted, as it only exists as a visual object, without any dependencies.

Changing Mesh - Button Object Replace Edition

(Pros: Cleaner Implementation, reduces total amount of gameObjects in the scene.
Cons: Easier to fail as the new buttonMesh will need to be manually assigned to the ButtonLogic script on the ButtonPrefab in the inspector).

1. Import your button mesh as a child of the ButtonPrefab and position it to align with the position and scale of the original ButtonMesh.
2. Delete the ButtonMesh Child object of the ButtonPrefab.
3. Select the ButtonPrefab gameObject in the scene/Hierarchy
4. Drag the new button mesh onto the “Button Mesh Object" exposed gameObject variable in the inspector under the Button Animation Properties tab on the Button Logic script in the inspector.
5. Drag and drop your custom 3D base mesh onto the ButtonPrefab object, and position and scale it accordingly.
6. The BaseMesh object can be deleted, as it only exists as a visual object, without any dependencies.

The 3 Method Unity Event Tabs

The buttons allow for 3 different states to be triggered; OnPress, OnStay, and OnLeave. This is to extend the functionality of the buttons as much as possible, whilst still keeping it simple and easy to use.

OnPress: Gets fired immediately upon the user pressing the button (I.e. the raycast hitting the button's collider. This only gets fired once per button click.

OnStay: Gets fired every frame that the user keeps his finger on the button

OnLeave: Gets fired once the user takes the finger off the button, by either repositioning the touch/mouse position to no longer be over the button, or ends the touch/mouse button press.

Tag Based Conditioning Instead of Component Based

This bundle also allows you to select a tag based check instead of component based, meaning that it will check if the button has a “button3D” tag, instead of checking if the gameObject has the ButtonLogic component attached to it.
To enable this, simply check the “Component Based Selection” bool in the inspector on the ButtonRaycastControllerGO off, and Set the tag of the “ButtonPrefab” gameObject to “button3D”.

Troubleshoot

The button doesn’t react at all?

1. Ensure the “ButtonRaycastController” script it attached to an active gameObject (such as the included "ButtonRaycastControllerGO”-prefab).
2. Check that the ButtonPrefab has the “button3D” tag assigned to it (The tag is Case-Sensitive, so copy paste button3D into the tag editor, if you’re unsure).
3. Ensure that that a collider is present on the ButtonPrefab object, and that the collider component is active and not set as a trigger.
4. Open the

Button Disappears on press but reappears again?

1. If the button has been resized from the prefab, ensure that the button isn’t traveling too far down, which can be tested by setting both the button down travel speed and total distance traveled variables found on the button prefab under the animation properties tab, to 0.00, and then simply increase these values until the desired result is present. If this doesn’t work, try moving the buttonMesh object further up prior to Playing the scene. As always, refer to the setup in the example scene to trouble shoot where the difference between your scene and the optimal setup is.

Acknowledgements

Thanks to GTRuiz for his Unity 2017 update of the Depth Text Shader - Accessible at https://forum.unity3d.com/threads/3d-text-that-takes-the-depth-buffer-into-account.9931/

Still Need Help? Fear not!

Contact me at **@AndersSchou** on **Twitter**, and I’ll try to sort out your problems, good luck!