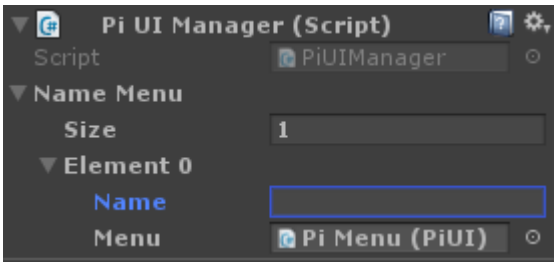
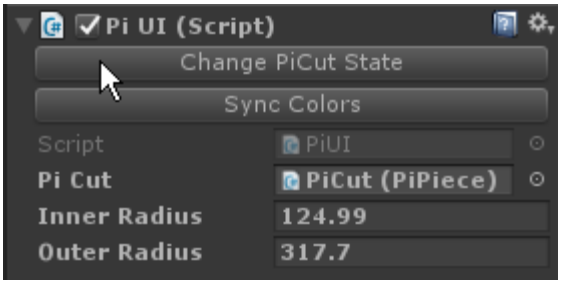
Pi Ui Documentation-

How To Use Pi Ui

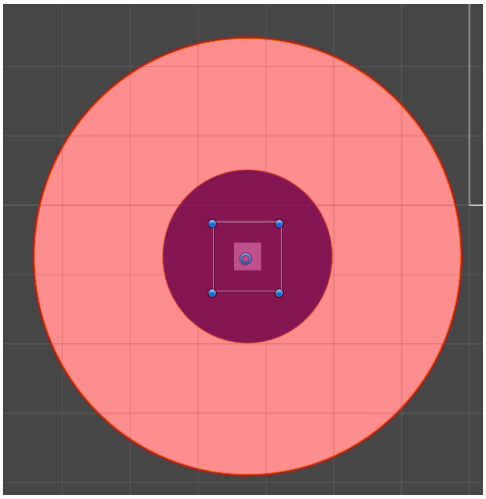
First to use Pi Ui one needs to have a “Pi Ui Canvas” object in the scene. To get this object simply go to the prefabs folder. Then drag and drop the “Pi Ui canvas” into the scene. Secondly the “Pi Ui Canvas” needs to have children “Pi Menu” objects. Following the previous steps one can grab the “Pi Menu prefab” and make it a child of the Canvas object. Returning to the “Pi Ui Canvas” object, the Pi UI manager will be visible in the inspector. As seen below each Pi Menu that’s a child to the canvas object will be added to a list with a visible name field. Which is the identifier to call the Pi Menu.



Following that go to the Pi Menu child object and click the “Change PiCut State” button. This will either activate or deactivate the PiCut. Allowing one to easily adjust the inner and outer radii.



Adjusting the above radii will expand/contract the gizmos. Adjust them until the gizmo matches the shape of the sprite on the PiCut object. If the shape is filled use an inner radius of 1 for optimal results. Following the adjustment of the radii click the “Change Pi Cut State” button. After that the remaining values of the Pi UI can be changed.



Pi Data can either be changed in code or manually using the Unity Inspector. Below will explain the basics of adjusting piData in code.

```
PiUI tempPi = piUi.GetPiUIOf("Normal Menu");
int i = 0;
foreach (PiUI.PiData data in tempPi.piData)
{
    data.sliceLabel = "Test" + i.ToString( );
    data.buttonActions = new UnityEngine.Events.UnityEvent( );
    data.buttonActions.AddListener(TestFunction);
    i++;
}
piUi.UpdatePiMenu("Normal Menu");
```

The script that is modifying the piData must have a reference to the “Pi Ui Canvas” object’s Pi Ui Manager. This can be got through code, or manually through the use of the [Serialized field] attribute on a private PiUiManager variable. Once there is a reference to the PiUiManager the Pi Ui Manager helper functions can be used.

In most cases getting the PiUi is the first step to modifying the piData. Typically a local variable is used to temporarily store it. Following that one can easily loop through the data in either a for loop or a for each loop. Modifying the PiData Variables to their desired values. If the PiUi.sliceCount , or piUi.equalSlices variables change use the PiUiManager.RegeneratePiMenu function. Otherwise the PiUiManager.UpdatePiMenu function is to be used.

```
piUi.ChangeMenuState("Normal Menu", new Vector2(Screen.width/2f,Screen.height/2f));
```

To adjust PiData manually click the Pi Menu object of the menu that is to be modified. Scroll down in the inspector and expand the pieData array. Then just simply drag your desired values to the open fields. Then later when the menu is called to change state it will open and have the previously set values.

Controller Support

Pi Ui’s controller support is simple to use, one must have reference to the menu that they want to interact with, then they simply assign that menu’s PiUI.joystickInput, and PiUI.joystickButton. An example of this is in the Test.cs file.

How To Expand PiUI

Pi Ui can be expanded in many ways the most likely reason for expansion is to add more transitional effects. To add these effects one must first add to the TransitionType enum

in PiUI. After that, adding the new enum to the switch statements in the Update, and OpenMenu functions . Lastly make the function. There are example functions located in the Transitions region in PiUI.

PiUiManager Functions

ChangeMenuState – Pass in a string that is the name of a menu, opening or closing the menu, if opening pass in the desired screen space position of where to open it.

PiOpened – Pass a string that is the name of the menu, will return the PiUI.menuOpened variable, indicating whether the menu is logically opened.

GetPiUIOf – Pass a string to get the PiUI of that menu allowing you to manually access the variables on PiUI.

RegeneratePiMenu – Passing a string that is the menu name. After changing PiUI.sliceCount or changing PiUi.equalSlices call this to remake the new slice count menu.

UpdatePiMenu – Passing a string that is the menu name. After only changing PiUi.PiData, call this to update the menu to the new PiData.

OverAMenu- If any menu under this PiUIManager detects input over a slice it will return true.This is useful for if one needs to raycast but does not want to when over a menu.

PiUi Functions

GeneratePi – Builds a menu at the passed position with PiUi.PiData settings.

UpdatePiUI – After only changing PiUi.Pidata, call this to update the menu to the new PiData.

OpenMenu – Opens the menu at the given position, if menu is not created it will create one.

SyncColor – Iterates through PiUi.piData changing the color’s to the sync ones in PiUi.

Transition Functions – Transitions when menu opens or closes, these can be added to/removed as long as the Switch statement in OpenMenu, Update and the enum PiUi.TransitionType is updated.

ResetPiRotation – Returns the slices to their appropriate rotation which is saved in PiUI.angleList.

PiRotationToNil – Returns slices to the rotation of Quaternion.Identity.

PiPieceFunctions

ManualUpdate – Called by PiUI to manage and reduce the amount of Monobehaviours using their own Update function.

Center – Returns the center of the slice, for placing the label and text

PiData Variables

angle – If the PiUi is not using equalSlices this will designate the angle the slice will be.

sliceLabel- This is the text that shows on the slice. PiPiece child at index 1. Change values on the Pi Menu's Pi Cut child to effect all on regeneration.

icon- This is the icon that shows on the slice. PiPiece child at index 0. Change values on the Pi Menu's Pi Cut child to effect all on regeneration.

IconSize- The width and height of the icon. Can be unique per slice.

nonHighlightedColor- This is the color of the slice when not selected.

highlightedColor- This is the color of the slice when selected.

disabledColor – This is the color of the slice when isInteractive is false.

onSlicePressed- Functions that will be called when the slice is interacted with.

hoverFunctions- This dictates whether this slice has hover functions enabled.

onHoverEnter/Exit- These functions fire once on their respective event either enter or exit.

order- The position in the menu starting from angle 0 this pi is, goes in anti clockwise order.

isInteractable- This will tell the slice whether it should detect inputs and respond to them, if a slice is disabled it will turn to the disabledColor.

PiData Function

SetValues- Pass in a piData that you want to copy the values of, and it will copy the values to the piData that called it.