Open Source VR Project

build passing API Documentation

OSVR User Interface Installation Guide

> Requires the Unity software version 2019.1 (or above).
>
> Requires the Oculus Integration plugin from the Unity Asset Store.

Getting Started

Documentation

Visit Full Documentation for a detailed API

Setting up the project

- Create a new project in the Unity software version 2019.1 using 3D Template or open an existing project.
- Ensure `Virtual Reality Supported` is checked:
    - In the Unity software select `Main Menu -> Edit -> Project Settings` to open the `Project Settings` window.
    - Select `Player` from the left hand menu in the `Project Settings` window.
    - In the `Player` settings panel expand `XR Settings`.
    - In `XR Settings` ensure the `Virtual Reality Supported` option is checked.
- Ensure the project `Scripting Runtime Version` is set to `.NET 4.x Equivalent`:
    - In the Unity software select `Main Menu -> Edit -> Project Settings` to open the `Project Settings` inspector.
    - Select `Player` from the left hand menu in the `Project Settings` window.
    - In the `Player` settings panel expand `Other Settings`.
    - Ensure the `Scripting Runtime Version` is set to `.NET 4.x Equivalent`.
- Download the `OSVR User Interface Framework` asset from the Unity Asset Store
    - Open the `UI/ExampleScenes/UIDemoScene` scene.

Getting Started

- Buttons
    - add a collider to the object you want to be a button
    - set the collider istrigger to True
    - add a button script to that object
    - set the activating and hovering layers to the layers you want them to interact with
    - ensure that the object can collide with these layers, you may need to check 'edit -> project settings -> physics' to see if the layers intersect
    - set the parameters to the desired number to change how the button reacts to hovering, if at all.
    - set the game objects that will be spawned based on the status of the button
    - set the unity events to have the correct delegates so they fire when the event occurs
- DropDowns
    - place a button script on the object you want to be a dropdown (follow the full guide above)
    - set the Hovering and activating layers in the dropdown
    - ensure that the object can collide with these layers, you may need to check 'edit -> project settings -> physics'

- ensure that your chosen template has a button script and a text mesh
    - set template to the gameobject that will be spawned for every button the dropdown makes
    - set Change In Height to an appropriate value for the spacing of the template between other buttons
    - set first offset to an appropriate value for the template to be spawned away from the main dropdown button
    - set Use Gradient or Use Instant Transition to true if you want all the buttons made by the drop down to have the same hovering effect
    - set the values below that to the values you want the spawned buttons to have when they are hovered.
    - if neither Use Gradient or Use Instant Transition are set to true the buttons will have the values specified on their option
    - go to options and set the number of options you want to display
    - set The label to the string you want to have displayed on the button's text mesh
    - set the parameters in option in the same manner as a button for all the extras gradients and colors
    - color fade of the gradient can only be set as the item fade in the dropdown script

- Keyboard

    - drag the keyboard prefab into your scene
    - the keys are made using our ButtonScript where the events firing are tied to script modifying a Text object
    - To set the keys' text object to the one you want to modify, click on the key or shift click and select multiple keys, and within the attached TextScript, select the text object you want to be modified by the keys
- Rotate To User
    - attach to the Game Object you want to rotate
    - set user camera to the camera the user uses to see the world
    - set is center to true if it should rotate in place if you set this to true ignore the rest of the set up guide
    - set the pivot point the object you want this object to orbit
    - set distance from pivot point to a reasonable value
- Scale to user
    - attach to the Game Object you want to scale
    - set user camera to the camera the user uses to see the world
    - set ScaleDist to the distance at which this object should be at its starting size
    - set near scale and far scale to the appropriate value for how quickly you want them to scale
    - set is center to true if it should scale based on its current position if you set this to true ignore the rest of the set up guide
    - set the pivot point the object you want this object to do the distance scaling based on
- Disappear At Distance
    - attach to the Game Object you want to toggle the visibility of
    - set user camera to the camera the user uses to see the world
    - set Disappear at Close Distance to a positive number if you want it to disappear when you get within that distance
    - set Disappear When Far Away and Disappear At Far Distance to True and the right value if you want it to disappear at a certain distance
    - set Item Renderer and Child Renderer to true if you want to have the renderer of the item and or the children of the item affected
    - set is center to true if it should scale based on its current position if you set this to true ignore the rest of the set up guide
    - set the pivot point the object you want this object to do the distance scaling based on