VR Keyboard setup instructions

VRKeyboard prefab contains VR Input Settings for interaction control method setup.

Gaze setup

Gaze interaction method use center of Canvas' Event Camera as a pointer. This method require Canvas for interaction with keyboard.

1. Create new Canvas and move it in your Main Camera prefab.
2. Add *VRKeyboard* prefab.
3. Add *VRInputSetiings* prefab.
4. Set the Control method to **GAZE**.
5. Put your Canvas in *GazeCanvas* required field.

   [Optional]

   - You can add *GazePointer* prefab to your Canvas and put it in Gaze PorgressBar. It will show interaction progress.
   - GazeClickTimer: Waiting time start from enter event and before press state cause.
   - GazeClickTimerDelay: Time for GazeClickTimer reset after press state caused.

****Gaze Demo** scene has been set up before for quick start.***

Google VR setup

1. Download Google VR SDK for Unity https://developers.google.com/vr/unity/download and import *.unitypackage* into the project.
2. Add *GoogleVR* prefabs you want to use in the scene.
3. Add *VRKeyboard* prefab.
4. Add *VRInputSetiings* prefab.
5. Set the Control method to **GOOGLEVR**.

****GoogleVR Demo** scene has been set up before for quick start.***

Oculus VR setup

If you want to use gamepad, oculus remote or oculus touch for keyboard interaction, you need to integrate Oculus SDK into the project:

1. Download *Oculus Utilities for Unity 5* https://developer3.oculus.com/downloads/ and import *.unitypackage* into the project.
2. Add *OVRCameraRig* prefab from Oculus Utilites package to your scene.
3. Add *VRKeyboard* prefab.
4. Add *VRInputSetiings* prefab.

There are two ways to interact with keyboard – use remote control or gamepad with gaze pointer system and use oculus touch.

Oculus remote and gamepad implementation:

- Set VR Input control method to **OCULUS_INPUT**.
- Press the “Enable” button to recompile for this control method.

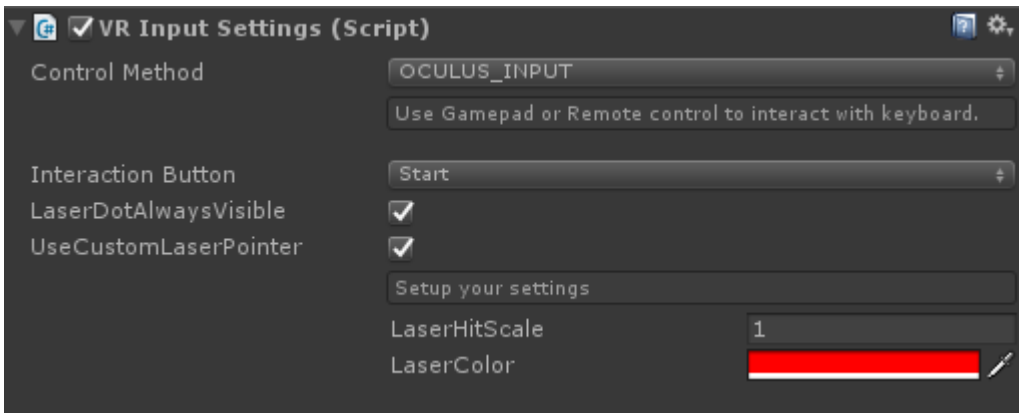
After that, additional settings will be available.

[Required]

- *InteractionButton*: Setup enum of interaction buttons. More info at https://developer.oculus.com/documentation/unity/latest/concepts/unity-ovrinput/ .Start button as default.

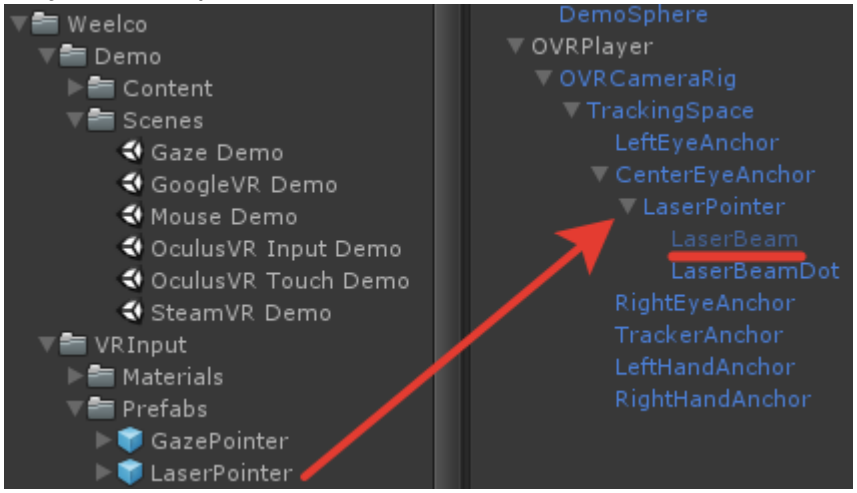
[Optional]

- *LaserDotAlwaysVisible*: If pointer is active, it’s trigger LaserPointer dot visible while focus on Keyboard.
  UseCustomLaserPointer: Automatically create laser pointer dot in CenterEyeAnchor gameobject in Play Mode. You can configure laser hit scale and laser color. Instead *CustomLaserPointer* you can add *LaserPointer* prefab to Touch controller gameobject. It will be your laser pointer as well.



[Additional]

- Instead *CustomLaserPointer* you can add *LaserPointer* prefab to CenterEyeAnchor gameobject and deactivate LaserBeam gameobject. It will be your laser pointer as well.
-



****OculusVR Input Demo* scene has been set up before for quick start. Just press the “Enable” button in *VRInputSettings* to recompile for this control method.****

Oculus touch implementation:

- Set VR Input control method to **OCULUS_TOUCH**.
- Press the “Enable” button to recompile for this control method.

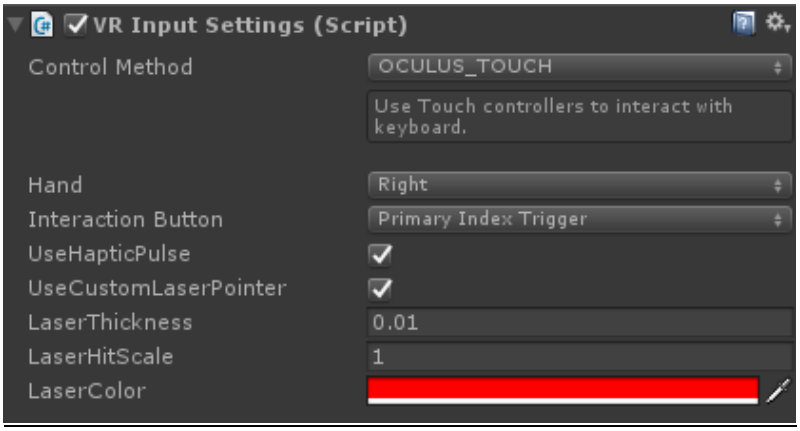
After that, additional settings will be available.

[Required]

- *InteractionHand*: Right, Left or Both.
- *InteractionButton*: Button to interact with keyboard. More info at
- https://developer.oculus.com/documentation/unity/latest/concepts/unity-ovrinput/ . Primary Trigger as default.

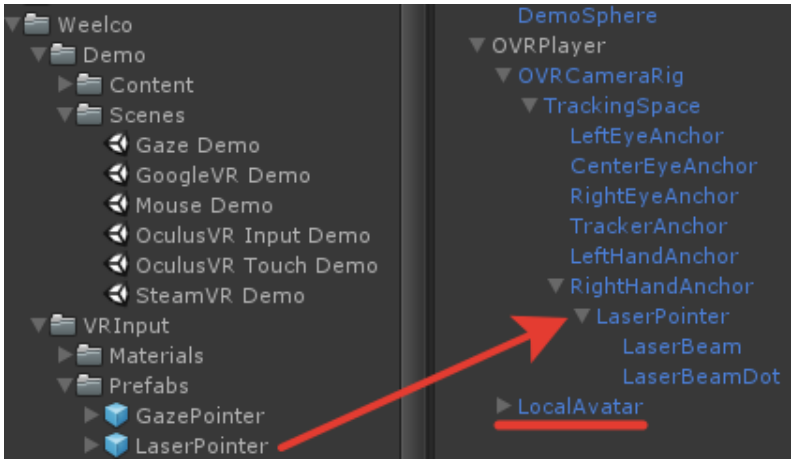
[Optional]

- *UseHapticPulse*: Cause haptic pulse while keyboard buttons Enter and Exit events.
- *UseCustomLaserPointer*: Automatically create laser pointer in selected hand in Play Mode. You can configure laser thickness, laser hit scale and laser color.



[Additional]

- Import *OvrAvatar* package https://developer.oculus.com/downloads/package/oculus-avatar-sdk/ into the project and add *LocalAvatar* prefab to your scene. Then you will see the touch controllers’ models in the scene.
- Instead *CustomLaserPointer* you can add *LaserPointer* prefab to Touch controller gameobject. It will be your laser pointer as well.



****OculusVR Input Demo* scene has been set up before for quick start. Just press the “Enable” button in *VRInputSettings* to recompile for this control method.****

Steam VR setup

1. Download *SteamVR Plugin* from Assets Store. https://www.assetstore.unity3d.com/en/#!/content/32647 and import into the project.
2. Add *[CameraRig]* prefab from *SteamVR* to the scene.
3. Add *VRKeyboard* prefab.
4. Add *VRInputSetiings* prefab.
5. Set its Control method to **VIVE**.
6. Press the “Enable” button to recompile for this control method.

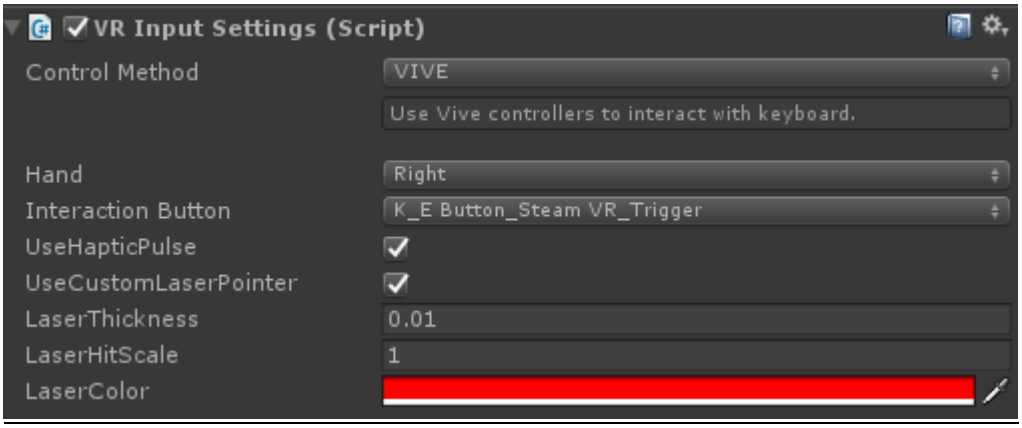
After that, additional settings will be available.

[Required]

- *InteractionHand*: Right, Left or Both.
- *InteractionButton*: Button to interact with keyboard. More info at https://docs.unity3d.com/Manual/OpenVRControllers.html. Steam VR Trigger as default.

[Optional]

- *UseHapticPulse*: Cause haptic pulse while keyboard buttons Enter and Exit events.
- *UseCustomLaserPointer*: Automatically create laser pointer in selected hand in Play Mode. You can configure laser thickness, laser hit scale and laser color.



[Additional]

- Instead *CustomLaserPointer* you can add *LaserPointer* prefab to Vive controller gameobject. It will be your laser guide as well.

****SteamVR Demo* scene has been set up before for quick start. Just press the “Enable” button in *VRInputSettings* to recompile for this control method.****

VR Keyboard settings

1. At first you need to initialize keyboard:

```
void Start() {
     keyboard.Init();
     keyboard.OnKeyPressed += OnKeyPressed;
}

void OnDestroy() {
     if (keyboard) {
         keyboard.OnKeyPressed -= OnKeyPressed;
     }
}
```

2. Then you need to handle input values :
   All keyboard layouts contained in `VRKeyboardData` class.

```
private void OnKeyPressed(string value) {
     if (value.Equals(VRKeyboardData.BACK)) {
         BackspaceKey();
     }
     else if (value.Equals(VRKeyboardData.ENTER)) {
         EnterKey();
     }
     else {
         TypeKey(value);
     }
}
```

For example backspace will remove last character in your text:

```
private void BackspaceKey() {
     if (text.Length >= 1)
         text = text.Remove(text.Length - 1, 1);
     }
 }
```

In other case character will be added in your text:

```
private void TypeKey(string value) {
     char[] letters = value.ToCharArray();
     for (int i = 0; i < letters.Length; i++) {
         text += letters[i].ToString();
     }
 }
```