VR Keyboard in Unity

Although different SDKs offer built-in keyboards, most don't offer cross-platform support. Below is an updated version of a VR friendly keyboard that has been modified from a now inactive blogger "Tales from the Rift." I have an included an updated version of their package including TextMeshPro elements.

The blog post instructions are still valid.

Download HERE - https://drive.google.com/file/d/1AoK9kyOT2Kru0hJb4C0DA4ho_fes2Skc/view?usp=drive_link

VR Canvas Keyboard

Hello Virtual World

I've received a few emails about inputting text in VR, specifically around Unity's Input Field.

Firstly, bugs, Input Fields rendered in world-space have several known bugs with displaying the caret in the wrong position. I'm not sure why Unity has not resolved these yet.

Secondly, conceptually text input in VR is obviously not going to be do-able with a physical keyboard, so you need a virtual input panel controllable by either a joystick input device and/or a virtual pointer. At this time though, Unity only supports input panels for mobile builds with a touch screen (iOS/Android) which are rendered in screen-space. Unfortunately Unity has no world-space keyboard listed on their roadmap.

In this blog, I'll show you how to create a canvas world-space keyboard for VR using look/gaze based control.

Watch the video below, when the keyboard is displayed the user can look at the keyboard button they want to trigger and select it with a physical button (Space on the PC physical keyboard, or X on the X-Box controller, or GearVR short-press button).

https://player.vimeo.com/video/137138875

It works well enough when you only have a small amount of input, such as getting the users name before they join a game, but I'd hate to use it in a vr chat scenario!

Quick Start

For those in a rush, here is how you can add a keyboard to your VR application:

1. Download the **.unitypackage** and load it into your project.

2. Create your text panel for input. To keep it simple, create a UI Button, and a canvas will automatically be created. Set that canvas to be world space, then scale and position it in the scene.
Note: We'll use a standard text button because of the bugs with Input Fields.

CanvasButton

3. Add an XR rig with an input mechanism, such as a ray interactor from the XR Interaction Toolkit.

Note: You can use an alternative input module, or even the standard touch input module. Whatever you like!

Gaze Event System

4. Create another canvas game object, also set in world-space which be the anchor for our actual keyboard. By default these are huge, so set it's scale to something meaningful like (0.002,0.002,0.002):

Canvas Keyboard Anchor

5. Lets wire up the click event on your button to open the keyboard on that canvas anchor.

Drag on the OpenCanvasKeyboard script to the button. And assign your Canvas from previous step 4 to the canvas property. Then assign the Text game object child to the Text attribute.

Finally, add an OnClick event handler to open the keyboard. Press + to add a handler, drag the button itself onto the Name (Object), and then in the drop down choose OpenCanvasKeyboard -> OpenKeyboard

OpenKeyboard

That's it. Now hit run. When you look towards the button and press SPACE the keyboard will display (you may need to rotate your canvas) and you should be able to type and see text change on the button.

Hello Virtual World

When you change scenes the keyboard is automatically destroyed, or if you want to manually close the keyboard call: CanvasKeyboard.Close();

Unity Canvas

Since Unity 4.6 and 5, Unity has introduced it's new UI system based on a 2d flat canvas that can be rendered to either screen or world-space. It includes all the dynamic sizing you would expect of a modern UI system. My keyboard is created out of UI buttons laid onto several panels, that are then attached to a canvas.

One of the best ways to create the keyboard is to initially lay it out using screen-space. Then you can more easily see how it scales.

If you see the screen shot below, I've used the Canvas Scaler to scale with screen size and used a reference pixel resolution of 1024×768, but I actually only care about the width (see how I've set Match slider to be width). Now now matter how wide the canvas scales it will always be 1024 reference pixels. The important part (which is not shown below) is that the panel "CanvasKeyboard" attached to the canvas is set to a width and height of 1024 x 400. So now we have a rectangular panel that scales maintaining it's aspect ratio.

Canvas Keyboard Screen Space

Important Note: If you are in VR Supported mode, and hit run, no screen space UI is displayed by design, as screen space UI doesn't make sense in VR. So when you are designing and testing a keyboard (or menu, or whatever) turn off VR Supported mode, and use the standard event system for mouse input.

There are many styles of keyboard, from an mobile device style with multiple panels, or to a single panel, or to funky round layouts (which no doubt work better in VR as they have less movement):

I have chosen a standard mobile input style keyboard. I did this because our poor VR users are already dealing with a lot to learn about such a new environment, that I wanted them to see a keyboard they recognize and are comfortable with – even if it is not as efficient as it could be. (I like think about the lesson Apple learnt trying to introduce handwriting recognition with the Apple Newton – when the iPhone was launched they kept their keyboard very standard!).

Laying out rows of buttons is a perfect match for combing the Canvas VerticalLayoutGroup (rows) and then each row using a HorizontalLayoutGroup (columns). I didn't use a grid layout because I wanted to stagger the buttons like a physical keyboard. I also wanted to have some different widths on keys such as SPACE and SHIFT. I can enforce the row heights by attaching a LayoutElement to each key and setting the Min Height property.

I did not use any padding between the keyboard buttons (even though it looks like it) because it made the keyboard a lot harder to use when there was a chance you were looking in between

buttons. So I faked it. I included space around the button background image in standard (unpressed) mode, and removed the padding in the high (pressed) mode. That way as you look at a button, it grows giving the illusion it has activated. Here are the background image:

keyboard background images

To use the keyboard, simply create a Canvas, drag the CanvasKeyboard prefab onto it as a child, and set the inputObject reference for the input object you want to edit. (add the Gaze/Look based VR input system , or use the mouse). Done!

A typical Canvas in world space looks like this:

canvas world space

Create the default canvas is MASSIVE. So for VR the scale x,y,z will be something like 0.002,0.002,0.002.

Keyboard key events

I am anticipating several keyboard layouts would be available (eg: Alphabetical, Numeric, URL, etc), so I've created a generic CanvasKeyboard object, and then specific Keyboard layouts below that, such as CanvasKeyboardASCII.

So each key is a button, and when it's OnClick occurs it calls CanvasKeyboardASCII.OnKeyDown(this). The CanvasKeyboardASCII then reads the key value from the name of the game object. Some keys it handles itself (such as close, or switching the panels), otherwise it sends the keypress up to the CanvasKeyboard.SendKeyString(). The role of the CanvasKeyboard class it to then set the text on the currently active game object or UI element. It does this by using reflection to see if the target object has a text property and if so updating it, otherwise it falls back to setting the target game objects name.

Pretty simple really!

CanvasKeyboard API

Dragging on prefabs is all well and nice if you have only one field to edit, but what if you have multiple? We need an API!

Unity has an excellent mobile keyboard accessible via the TouchScreenKeyboard API. I really like the simplicity of that for keyboard management, so I created a very similiar one for my

CanvasKeyboard. That makes it a fairly straight forward transition if you already use the TouchScreenKeyboard.

They Canvas Keyboard can be opened and closed via static functions:

```
CanvasKeyboard Open(Canvas canvas, GameObject inputObject = null, CanvasKeyboardType keyboardTyp
```

Open the keyboard and parent it to a canvas. You should create your canvas as previously, and pass a reference to it here. Also add your input object that you want to keyboard to set text of. You can only have one canvas keyboard open, so if you already have one opened it will be closed.

```
void Close()
```

Close a keyboard if it is open.

```
bool IsOpen
```

Check if a keyboard is currently opened (the user may have closed it with the close button).

Here is the code for managing it:

CanvasKeyboard.cs

```
public class CanvasKeyboard : MonoBehaviour
{
    public enum CanvasKeyboardType
    {
        ASCIICapable
    }

    public static CanvasKeyboard Open(Canvas canvas, GameObject inputObject = null, CanvasKe
    {
        Close();
        CanvasKeyboard keyboard = Instantiate<CanvasKeyboard>(Resources.Load<CanvasKeyboard>
        keyboard.transform.SetParent(canvas.transform, false);
        keyboard.inputObject = inputObject;
        return keyboard;
    }

    public static void Close()
    {
        CanvasKeyboard[] kbs = GameObject.FindObjectsOfType<CanvasKeyboard>();
        foreach (CanvasKeyboard kb in kbs)
        {
            kb.CloseKeyboard();
        }
    }

    public static bool IsOpen
```

```
        {
            get
            {
                return GameObject.FindObjectsOfType<CanvasKeyboard>().Length != 0;
            }
        }
    }
```

Revision #2
Created 2024-04-22 13:38:30 UTC by Wes
Updated 2024-07-12 13:49:13 UTC by Wes