Interactive World Space UI

Button controlled water hose

Creating a dashboard with buttons

Create a dashboard with a Start and a Stop button:

1. From the Project panel, drag the DefaultCanvas prefab onto the MeMyselfEye object in the Hierarchy panel so that it becomes a child.

2. Rename it to Dashboard.

3. With Dashboard selected, set the Rect Transform component's Pos X, Pos Y, Pos Z to (0, -1, 0.2) and its Rotation to (60, 0, 0).

4. Disable or delete the Text child object of Dashboard.
This places the dashboard one meter below our eyes and a little out in front.

To include an image sketch of a vehicle dashboard:

1. Import the DashboardSketch.png file into your Project (such as the Assets/Textures folder).

2. Add a new GameObject | UI | Raw Image as a child of Dashboard.

3. Drag the DashboardSketch texture from the Project panel onto the Texture field of the Raw Image component.

4. Set its Rect Transform component's Pos X, Pos Y, Pos Z to (0,0,0), Width to 140, and Height to 105.

5. It should be Anchored at middle-center (0.5,0.5) in X, Y, and Pivot, with Rotation (0,0,0).

6. Set Scale to (4.5,4.5,4.5).

To add the Start and Stop buttons:

1. Add a new GameObject | UI | Button as a new child of Dashboard. Name it StartButton.

2. Set its Rect Transform component's X, Y, Z to (-48, 117, 0), ser Width, Height to (60, 60), and Anchored to center-middle (0.5). No Rotation and Scale of 1.

3. In the button's Image (Script) component pane, for Source Image, click on the tiny circle on the far right to open the Select Sprite picker and choose ButtonAcceleratorUpSprite (which you may have imported into the Assets/Standard Assets/CrossPlatformInput/Sprites folder).

4. In the button's Button (Script) component pane, for Normal Color, RGB (89,154,43) and set Highlighted Color to (105, 255, 0).
5. Similarly, create another button named StopButton with the Rect Transform component's X, Y, Z (52, 118, 0), set Width, Height to (60, 60), for Source Image select ButtonBrakeOverSprite, choose Normal Color (236, 141, 141) and Highlighted Color (235, 45, 0).

To use the ReticleCursor created earlier with the *CursorPositioner.cs* script, the dashboard itself needs to have a collider for the script. We can achieve this by performing the following steps:

1. With Dashboard selected, right-click for options, and navigate to 3D Object | Plane.

2. Set its Position to (0,0,0), Rotation to (270,0,0) and Scale to (64,1,48).

3. Disable its Mesh Renderer (but leave its Mesh Collider enabled).

Now the dashboard has a plane child that isn't rendered, but its collider will be detected when CursorPositioner does its ray cast.

Linking the water hose to the buttons

1. If you haven't done so already, import the Particle Systems standard asset from the main menu bar and navigating to Assets | Import Package | ParticleSystems.

2. In the Project panel, find the Assets/Standard Assets/Particle Systems/Prefabs/Hose prefab and drag it into the Scene view.

3. Set its Transform component's X, Y, Z to (-3, 0, 1.5) and Rotation to (340, 87, 0).

4. Ensure that Hose is enabled (check its Enable checkbox).

5. Unfold the Hose in Hierarchy so that you can see its child WaterShower particle system. Select it.

6. In Inspector, in the Particle System properties pane, look for Play On Awake and uncheck it.

7. The Hose prefab itself comes with mouse-driven script that we don't want to use, so disable it:
   With Hose selected, disable (uncheck) its Hose (Script).
   Also disable (uncheck) the Simple Mouse Rotator (Script) component.

8. Wire up StartButton to the WaterShower particle system, by telling the buttons to listen for the OnClick() events:

- Unfold the Hose in Hierarchy so that you can see its child WaterShower particle system.
- In Hierarchy, select StartButton (under MeMyselfEye/Dashboard).
- Note that the On Click() pane in the Inspector is empty. Click on the *Plus* (+) icon on the lower right of that pane to reveal a new field labeled None (Object).
- Drag the WaterShower particle system from Hierarchy onto the None (Object) field.
- Its function selector the default value is No Function. Change it to ParticleSystem | Play().

9. The steps are similar for the StopButton.

- In Hierarchy, select StopButton.
- Click on the *Plus* (+) icon on the lower right of its On Click() pane.
- Drag the WaterShower from Hierarchy onto the None (Object) field.
- Its function selector the default value is No Function. Change it to ParticleSystem | Stop().

Play test your project. You should be able to click the Start/Stop button to fire and stop the water hose.

Activating buttons from the script

Before we give the user a way to press the buttons, let's see how we can do this from a script. Create a new script named *ButtonExecuteTest.cs* and attach it to GameController, In the script, we turn the hose on and off in five second intervals.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class ButtonExecuteTest : MonoBehaviour {

    public Button startButton, stopButton;
    private bool on = false;
    private float timer = 5.0f;

        // Use this for initialization
        void Start () {
         startButton = GameObject.Find("StartButton").GetComponent<Button>();
         stopButton = GameObject.Find("StopButton").GetComponent<Button>();
        }

        // Update is called once per frame
        void Update () {

         timer -= Time.deltaTime;
         if (timer <0.0f)
         {
             on = !on;
             timer = 5.0f;
         }

         if (on)
         {
```

```
            //water hose fires
            startButton.onClick.Invoke();
            startButton.OnPointerEnter(new PointerEventData(EventSystem.current)); //hightlighted
        }
        else
        {
            //water hose stops
            stopButton.onClick.Invoke();
            startButton.OnPointerExit(new PointerEventData(EventSystem.current)); //normal color

        }

    }
}
```

Look to highlight a button

Although Button is a Unity UI object, it needs to be detected with a ray cast. We will add a game object sphere to each button, tag it as Button and cast a ray to detect it.

1. In the Hierarchy panel, select StartButton (under MeMyselfEye/ Dashboard), right-click for options, and navigate to 3D Object | Sphere.

2. Set its Transform component's Scale to (52, 52, 52) so that it fits the button size.

3. To add a tag named Button, near the top of the Inspector panel, click on the Tag selector, click on Add Tag, and then select the Plus (+) icon to add a row to the tag list. Enter the name Button in place of New Tag.

4. We're tagging the sphere child of StartButton. So, click on the sphere in Hierarchy again so that its Inspector panel is visible again. Then, click on the Tag selector. This time, Button should be included in the list. Select it.

5. Disable the sphere's Mesh Renderer by unchecking the Mesh Renderer checkbox.

6. Repeat these steps for StopButton.

7. Create a new script *ButtonExecute.cs* (see code on **Canvas**) on GameController, **disable** the previous *ButtonExecuteTest* script now by unchecking its Enable checkbox. Use raycast as:

```
        Transform camera = Camera.main.transform;
        Ray ray= new Ray(camera.position, camera.rotation * Vector3.forward);
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit))
        {
            if (hit.transform .gameObject .tag =="Button")
            { …
            }
    …
        }
```

Head gestures controlled UI

Using the head position

1. Let's try just using the camera angle to tell if you're looking down, say within 60 degrees of looking straight down. Create a new script on the GameController named *HeadGesture.cs*, which checks whether you're facing down.

```
float cameraAngleFromGnd = Vector3.Angle(Vector3.down,
Camera.main.transform.rotation * Vector3.forward); the camera angle
with respect of straight down.
```

2. Create another new script also on the GameController and name it *FlippinDashboard.cs*, which flips opens the dashboard when you're looking down,

Using the head gesture

Instead of opening the dashboard by simply looking down, the user must look down quickly with the intent in the gesture of looking down. In other words, if you casually look down, the dashboard doesn't open. If you look down quickly, it does - a simple example of a head gesture input.

```
private bool DetectMovingDown () {
float angle = CameraAngleFromGround ();
float deltaAngle = previousCameraAngle - angle;
float rate = deltaAngle / Time.deltaTime;
previousCameraAngle = angle;
return (rate >= sweepRate);
}
```

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class ButtonExecute : MonoBehaviour {

    //public Button startButton, stopButton;
    Button hitButton, currentButton;

    private bool on = false;
    private float timer = 5.0f;

    // Use this for initialization
    void Start()
    {
       // startButton =
GameObject.Find("StartButton").GetComponent<Button>();
        //stopButton = GameObject.Find("StopButton").GetComponent<Button>();
    }

    // Update is called once per frame
    void Update () {

        Transform camera = Camera.main.transform;
        Ray ray= new Ray(camera.position, camera.rotation * Vector3.forward);
        RaycastHit hit;
        //GameObject hitObject;

        if (Physics.Raycast(ray, out hit))
        {
            if (hit.transform .gameObject .tag =="Button")
            {
                hitButton =
hit.transform.parent.gameObject.GetComponent<Button>();
                print("name= "+ hitButton.name);
            }

            if (currentButton != hitButton)
            {
                //unhighlight

                if (currentButton != null) { currentButton.OnPointerExit(new
PointerEventData(EventSystem.current)); }


                //make changes
                currentButton = hitButton;
                if (currentButton != null) {
                    currentButton.onClick.Invoke();
                    currentButton.OnPointerEnter(new
PointerEventData(EventSystem.current));
                }
            }

        }
    }
```

```
}
public class FlippingDashboard : MonoBehaviour {
    private HeadGuesture gesture;
    private GameObject dashBoard;
    private bool open = true;
    private Vector3 startRotation;


      // Use this for initialization
      void Start () {
        gesture = GetComponent<HeadGuesture>();
        dashBoard = GameObject.Find("MeMyselfEye/Dashboard");
        print(dashBoard.name);
        startRotation = dashBoard.transform.eulerAngles;
        print("startrotation" + startRotation);

      }

      // Update is called once per frame
      void Update () {

        if (gesture .isFacingDown )
        {
            OpenDashboard();
            print("open"+dashBoard.transform.eulerAngles);


        }
        else
        {
            CloseDashboard();
            print("close" + dashBoard.transform.eulerAngles);
        }

      }


    private void CloseDashboard()
    {
        if (open)
        {
            dashBoard.transform.eulerAngles = new Vector3(180.0f,
startRotation.y, startRotation.z);
            open = false;
        }
    }

    private void OpenDashboard()
    {
        if (!open)
        {
            dashBoard.transform.eulerAngles = startRotation ;
            open = true;
        }
    }
}
```

```
public class HeadGesture : MonoBehaviour {
public bool isFacingDown = false;
public bool isMovingDown = false;
private float sweepRate = 100.0f;
private float previousCameraAngle;
void Start () {
previousCameraAngle = CameraAngleFromGround ();
}
void Update () {
isFacingDown = DetectFacingDown ();
isMovingDown = DetectMovingDown ();
}
private float CameraAngleFromGround () {
return Vector3.Angle (Vector3.down, camera.transform.rotation
* Vector3.forward);
}
private bool DetectFacingDown () {
return (CameraAngleFromGround () < 60.0f);
}
private bool DetectMovingDown () {
float angle = CameraAngleFromGround ();
float deltaAngle = previousCameraAngle - angle;
float rate = deltaAngle / Time.deltaTime;
previousCameraAngle = angle;
return (rate >= sweepRate);
}
}
```

Add the isMovingDown detection to the FlippingDashboard script and replace Update(), as follows:

```
void Update() {
if (gesture.isMovingDown) {
      OpenDashboard ();
} else if (!gesture.isFacingDown) {
      CloseDashboard ();
}
}
```