World Space UI

In VR, we use various approaches to place user interface (UI) elements in world space instead of screen space. In the assignment of Gaze-based Control, you might have noticed that using screen space for UI in VR doesn't work. We will introduce the following different types of world space UI with examples.

- **Visor heads-up display**: In visor **heads-up display** (**HUD**) the user interface canvas appears at the same spot in front of your eyes regardless of the head movement

- **Windshield HUD**: This is a pop-up panel floating in 3D space like a windshield in a cockpit

- **Game element UI**: The canvas is in the scene as a part of the gameplay, like a scoreboard in a stadium

- **Reticle cursors**: Similar to visor HUD, a crosshair or a pointer cursor is used to choose things in the scene

- **Info bubble**: This is a UI message that is attached to objects in the scene, like a thought bubble hovering over a character's head

To start, we will prepare the following two prefabs.

Creating the MeMyselfEye prefab

1. From the main menu bar, navigate to **GameObject | Create Empty**. Rename the object MeMyselfEye
2. Set its position up close into the scene, at **Position** (0, 1.4, -1.5).
3. In the **Hierarchy** panel, drag the **Main Camera** object into MeMyselfEye so that it's a child object.
4. With the **Main Camera** object selected, reset its transform values
5. Drag the MeMyselfEye prefab into the **Project** panel, under Assets/Prefabs folder to create a prefab.

A reusable default canvas prefab

1. Create a new canvas and change its **Render Mode** to **world space** as follows:
   Navigate to **GameObject** | **UI** | **Canvas**.
   Rename the canvas as DefaultCanvas.
   Set **Render Mode** to **world space**.

2. In **Rect Transform**, set **Width** = 640 and **Height** = 480.

3. In **Scale**, set **X**, **Y**, **Z** to (0.00135, 0.00135, 0.00135).

4. **Rect Transform**, set **Pos X**, **Pos Y**, **Pos Z** to (0, 1.325, 0).

5. Add an empty **Image** element (with a white background) to help us visualize the otherwise transparent canvas and provide an opaque background for the canvas when we need one

   - With DefaultCanvas selected, navigate to **GameObject** | **UI** | **Image** (ensure that it's created as a child of DefaultCanvas; if not, move it under DefaultCanvas).

   - With the **Image** selected, on the upper left of its **Rect Transform** pane, there is an **anchor presets** button. Selecting it opens the **anchor presets** dialog box. Press and hold the *Alt* key to see the **stretch** and **position** options and choose the one on the bottom-right corner (**stretch-stretch**). Now, the (blank) image is stretched to fill the canvas

6. Add a **Text** element

   - With DefaultCanvas selected, navigate to **GameObject** | **UI** | **Text** (ensure that it's created as a child of DefaultCanvas,

   - With the **Text** selected, set **Alignment** to **Center Align** and **Middle Align** and set **Vertical Overflow** to **Overflow**. Set the **Scale** to (4, 4, 4).

   - set its **anchor presets** button to (**stretch** - **stretch**) using the widget on the upper left of its **Rect Transform** pane.

   - Increase the pixel resolution to give cleaner text fonts by keeping DefaultCanvas selected and setting the **Canvas Scaler** | **Dynamic Pixels Per Unit** to 10.

   - save it as a prefab asset

Visor HUD (heads-up display)

1. In the Hierarchy panel, unfold the MeMyselfEye object and then drill down to the Main Camera object

2. From the Project panel, drag the DefaultCanvas prefab onto the Camera object so that it becomes a child of Camera.

3. In the Hierarchy panel, with the canvas selected, rename the canvas to VisorCanvas.

4. In the Inspector panel for the canvas, change the Rect Transform component's Pos X, Pos Y, Pos Z to (0, 0, 1).

5. Unfold VisorCanvas and select the child Text object.

6. In the Inspector panel, change the text from Default Text to Welcome to Virtual Reality.

7. Change the text color to something bright, such as green.

8. Disable the Image object so that only the text shows, by unchecking its **Enable** checkbox in **Inspector**.

9. Save the scene, and try it in VR.

The windshield HUD

A visor HUD is like it’s attached to your head. A windshield HUD is like attached to your seat/windshied.

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

2. Rename it to HUDCanvas.

3. With HUDCanvas selected, set the Rect Transform component's Pos X, Pos Y, Pos Z to (0, 0.4, 0.8).

4. Now, we'll set the Text component. With Text under HUDCanvas selected, change the text to Welcome! Virtual Reality. Also, change the color to something bright, such as green.

5. This time, we'll make the panel translucent. Select the image from Image under HUDCanvas and select its color swatch. Then in the Color dialog, modify the Alpha ("A") channel from 255 to about 115.

The game element UI

1. From the **Project** panel, drag the DefaultCanvas prefab directly into the **Scene** view

2. Rename it ScoreBoard.

3. With ScoreBoard selected, set the **Rect Transform** component's **Pos X**, **Pos Y**, **Pos Z** to (-2.8, 7, 4.9) and **Width**, **Height** to (3000, 480).

4. With **Text** under ScoreBoard selected, set **Font Size** to 100 and a noticeable color such as red for the **Text**.

5. Enter the **Score: 0** sample string for **Text**.

6. Disable **Image** under ScoreBoard by unchecking the **Enable** checkbox, or deleting it.

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

public class KillTarget : MonoBehaviour {
public GameObject target;
public ParticleSystem hitEffect;
public GameObject killEffect;
public float timeToSelect = 3.0f;
public int score;

public Text scoreText;
private float countDown;

void Start () {
     score = 0;
     countDown = timeToSelect;
     hitEffect.enableEmission = false;
     scoreText.text = "Score: 0";
}

…

// killed
…
score += 1;
scoreText.text = "Score: " + score;
…
```

The reticle cursor

1. From the **Project** panel, drag the DefaultCanvas prefab onto the camera object so that it becomes a child of the camera. Name it ReticleCursor.

2. Set the **Rect Transform** component's **Pos X**, **Pos Y**, **Pos Z** to (0, 0, 1).

3. Delete its child objects—**Image** and **Text**.

4. Add a raw image child by selecting from the main menu bar navigating through **GameObject | UI | Raw Image** and making sure that it's a child of ReticleCursor.

5. In the **Raw Image** panel's **Rect Transform**, set **Pos X**, **Pos Y**, **Pos Z** to (0, 0, 0) and **Width**, **Height** to (22, 22). Then, choose a noticeable **Color** such as red in the **Raw Image (Script)** properties.

6. Save the scene and try it in VR.

7. If you'd like a nicer looking reticle, in the **Raw Image (Script)** properties, populate the **Texture** field with a cursor image.

8. Any issues with this set up?

We set its **Pos Z** to 1.0 so that the reticle floats in front of you at a one meter distance. A fixed distance cursor is fine in most UI situations. However, this is world space. If another object is between you and the reticle, the reticle will be obfuscated. We will tackle this problems in the lab assignment.

The info bubble

1. From the **Project** panel, drag the DefaultCanvas prefab directly into the **Scene** view on top of WalkTarget under GameController so that it's a child of WalkTarget.

2. Rename it to InfoBubble.

3. With InfoBubble selected, set the **Rect Transform** component's **Pos X**, **Pos Y**, **Pos Z** to (0, 0.2, 0).

4. With **Text** under InfoBubble selected, set the **Rect Transform** component's **Pos X**, **Pos Y**, **Pos Z** to (0, 0, 0) and **Right**, **Bottom** to 0, 0.

5. With **Image** under InfoBubble selected, set **Scale** to (0.7, 0.2, 1).

6. Enter the **X:00.00, Z:00.00** sample string for **Text**.

7. Modify script *LookMoveTo.cs* to show the current WalkTarget X, Z position. We will work on this part in the lab assignment.