Research log

Papers looked into :

[1] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. Slicetype: fast gaze typing with a merging keyboard. Journal on Multimodal User Interfaces,
13:321 – 334, 2018.
[2] Gunnar Borg. Borg's Perceived Exertion And Pain Scales. 07 1998.
[3] Anke Huckauf and Mario H. Urbina. Gazing with peyes: Towards a universal input for various applications. In Proceedings of the 2008 Symposium
on Eye Tracking Research amp; Applications, ETRA '08, page 51–54, New York, NY, USA, 2008. Association for Computing Machinery.
[4] Anders S. Johansen, John P. Hansen, Dan W. Hansen, Kenji Itoh, and Satoru Mashino. Language technology in a predictive, restricted on-screen
keyboard with dynamic layout for severely disabled people. In Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods,
2003.
[5] Jiaye Leng, Lili Wang, Xiaolong Liu, Xuehuai Shi, and Miao Wang. Efficient flower text entry in virtual reality. IEEE Transactions on Visualization
and Computer Graphics, 28(11):3662–3672, 2022.
[6] Mathias N. Lystbæk, Ken Pfeuffer, Jens Emil Sloth Grønbæk, and Hans Gellersen. Exploring gaze for assisting freehand selection-based text entry in
ar. Proc. ACM Hum.-Comput. Interact., 6(ETRA), may 2022.
[7] I. Scott MacKenzie and R. William Soukoreff. Phrase sets for evaluating text entry techniques. In CHI '03 Extended Abstracts on Human Factors in
Computing Systems, CHI EA '03, page 754–755, New York, NY, USA, 2003. Association for Computing Machinery.
[8] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. Fast gaze typing with an adjustable dwell time. In Proceedings of the SIGCHI Conference on
Human Factors in Computing Systems, CHI '09, page 357–360, New York, NY, USA, 2009. Association for Computing Machinery.
[9] Jacek Matulewski and Mateusz Patera. Usability of the super-vowel for gaze-based text entry. In 2022 Symposium on Eye Tracking Research and
Applications, ETRA '22, New York, NY, USA, 2022. Association for Computing Machinery.
[10] Marco Porta and Matteo Turina. Eye-s: A full-screen input modality for pure eye-based communication. In Proceedings of the 2008 Symposium on
Eye Tracking Research amp; Applications, ETRA '08, page 27–34, New York, NY, USA, 2008. Association for Computing Machinery.
[11] David J. Ward, Alan F. Blackwell, and David J. C. MacKay. Dasher—a data entry interface using continuous gestures and language models. In
Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology, UIST '00, page 129–137, New York, NY, USA, 2000.
Association for Computing Machinery.
[12] Jacob O. Wobbrock, James Rubinstein, Michael W. Sawyer, and Andrew T. Duchowski. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In Proceedings of

Research log

the 2008 Symposium on Eye Tracking Research amp; Applications, ETRA ’08, page 11–18, New York, NY, USA, 2008.
[13] Andrew T Duchowski. A breadth-first survey of eye-tracking applications. Behavior Research Methods, Instruments, & Computers, 34(4):455–470, November 2002.
[14] Robert J.K. Jacob and Keith S. Karn. Commentary on section 4 - eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In J. Hyönä, R. Radach, and H. Deubel, editors, The Mind’s Eye, pages 573–605. North-Holland, Amsterdam, 2003.
[15] Samantha D. Aziz Aziz and Oleg V. Komogortsev. An assessment of the eye tracking signal quality captured in the hololens 2. May 2022.
[16] Robert J.K. Jacob and Keith S. Karn. Commentary on section 4 - eye tracking in human-computer interaction and usability research: Ready to
deliver the promises. In J. Hyönä, R. Radach, and H. Deubel, editors, The Mind’s Eye, pages 573–605. North-Holland, Amsterdam, 2003.
[17] Marco Porta and Matteo Turina. Eye-s: A full-screen input modality for pure eye-based communication. In Proceedings of the 2008 Symposium on Eye Tracking Research amp; Applications, ETRA ’08, page 27–34, New York, NY, USA, 2008. Association for Computing Machinery.
[18] Päivi Majaranta and Andreas Bulling. Eye Tracking and Eye-Based Human–Computer Interaction, pages 39–65. Springer London, London, 2014.

Choice of Hardware:

The design and study to be implemented with Unity and with the Microsoft Mixed Reality Toolkit (MRTK) on the Microsoft HoloLens 2. This Head-mounted Display has a resolution of 2048 * 1080 pixels per eye, 8MP camera, and 2 IR cameras for eye tracking.

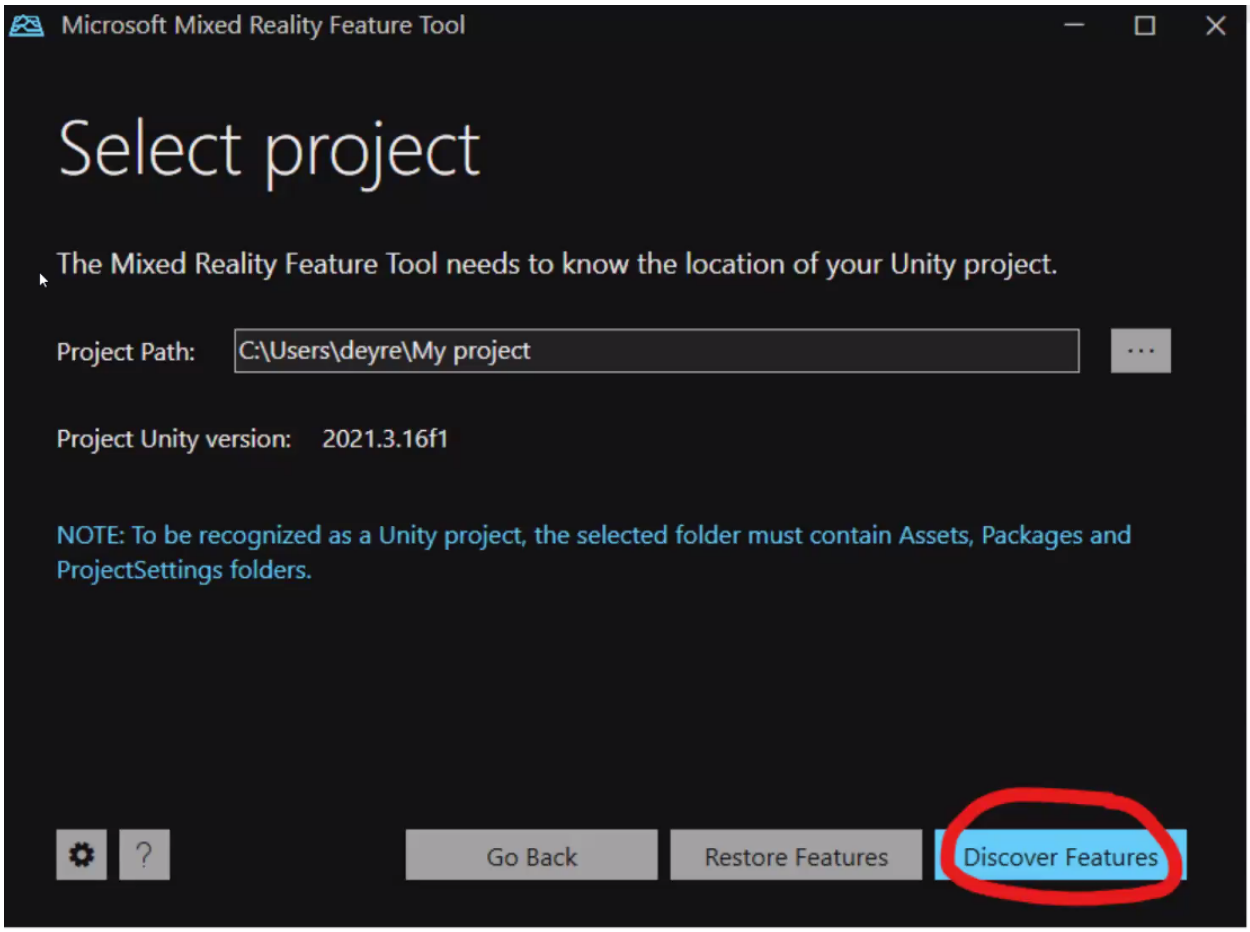
Set Up:

Unity Set up :

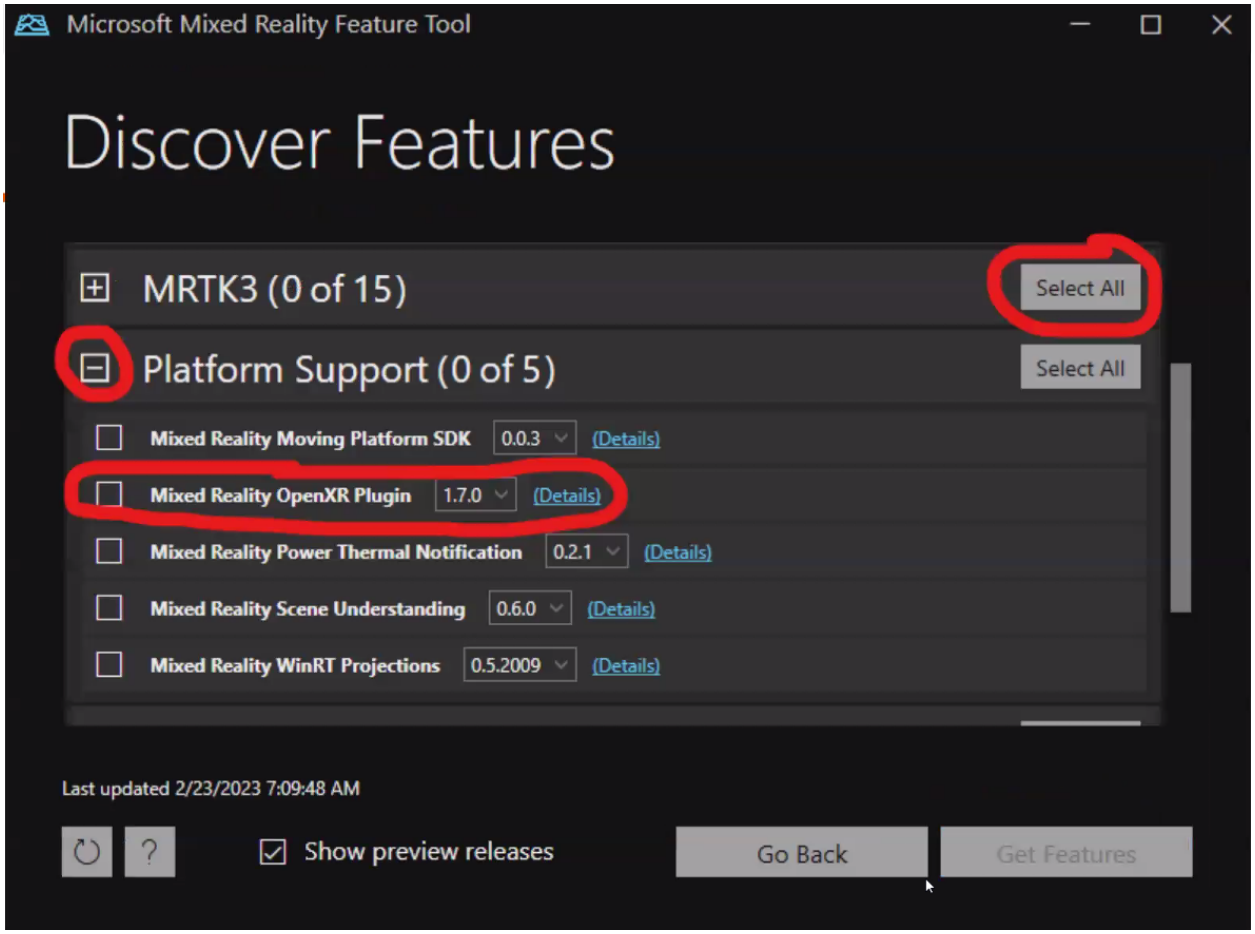
1) Download Unity hub
2) Choose Unity Version (2021.3.11.f1)
3) Install module UWP(Universal Windows Platform), Windows
4) Start new Project 3d core

Research log

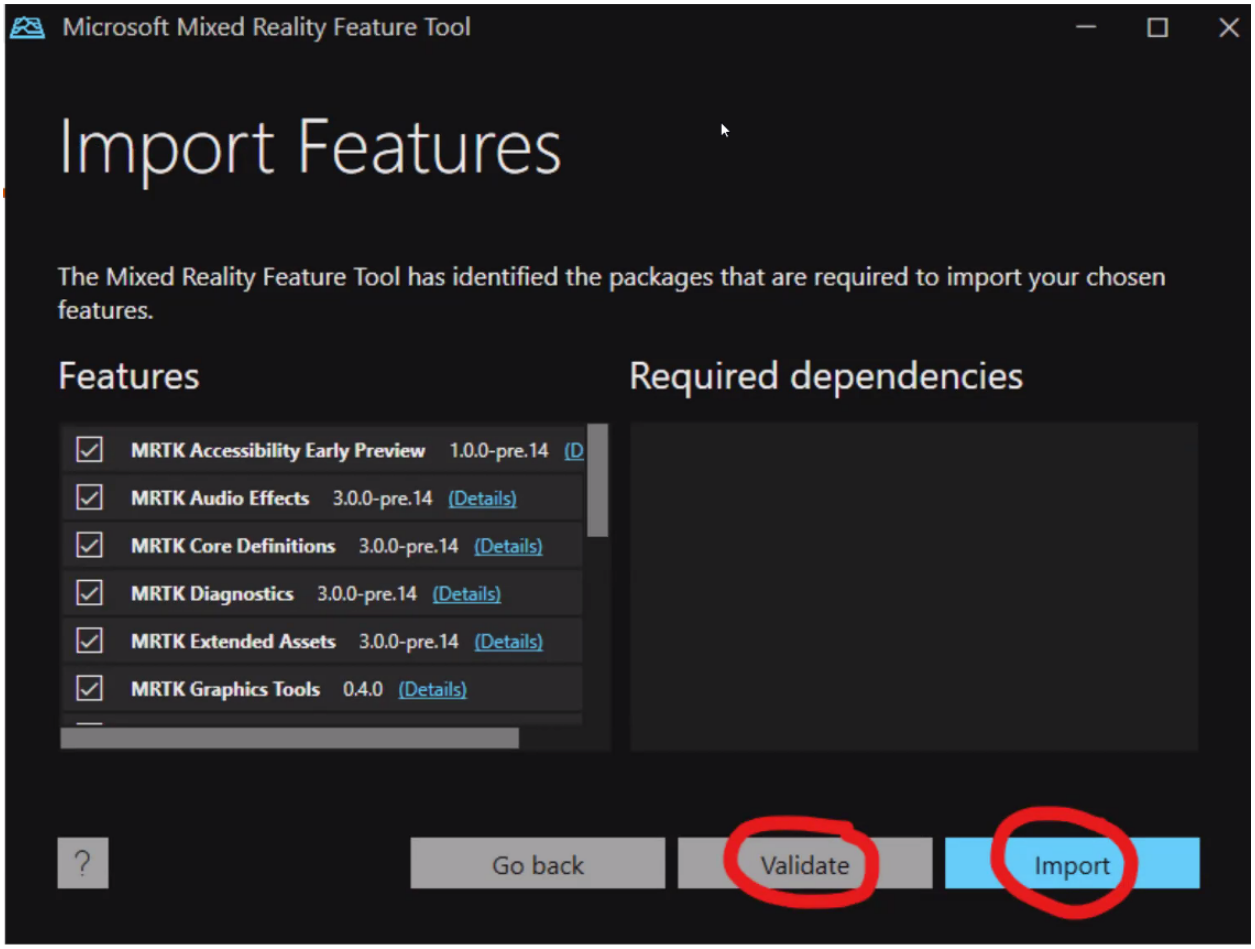
Mixed reality feature tool:



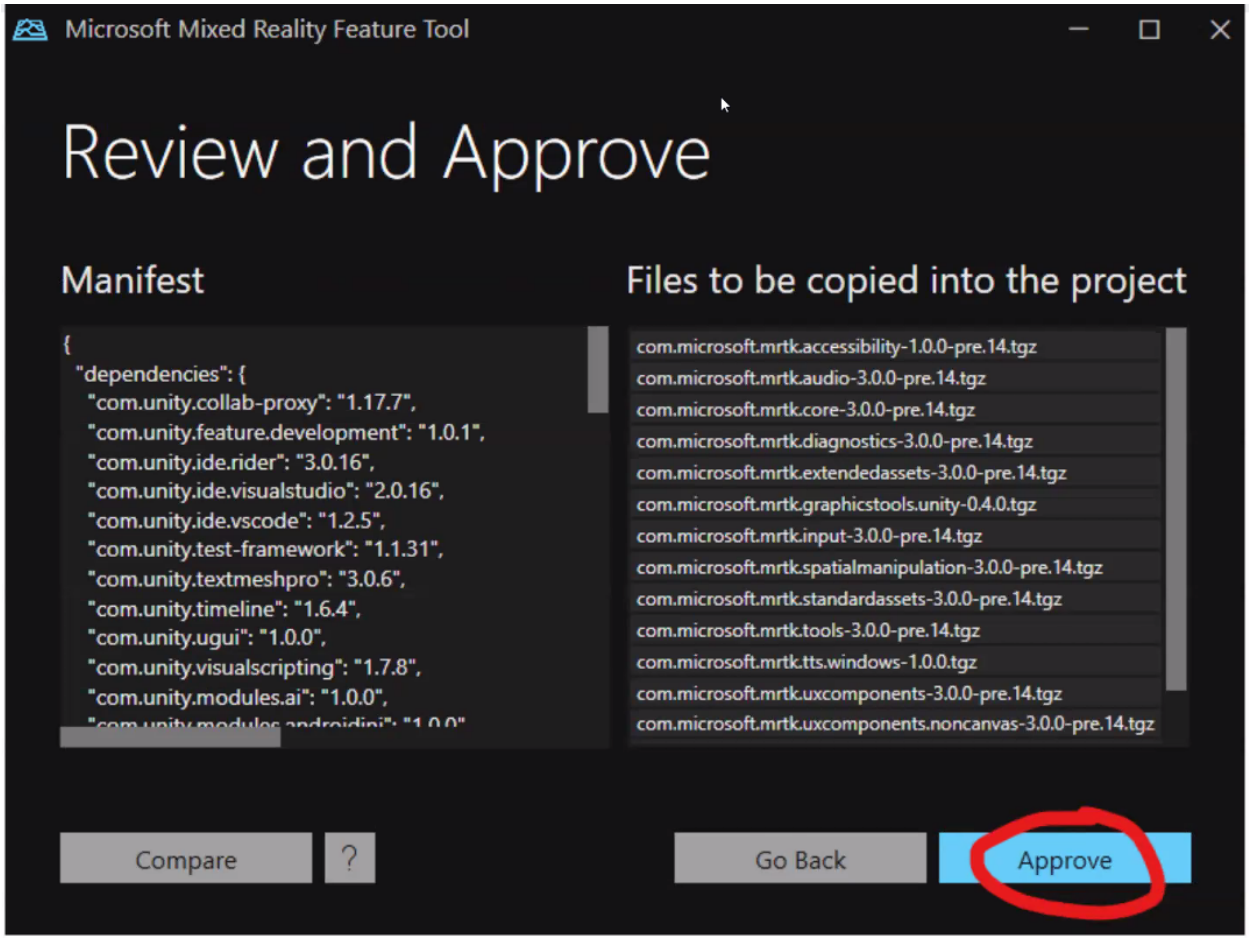
Research log



Research log



Research log

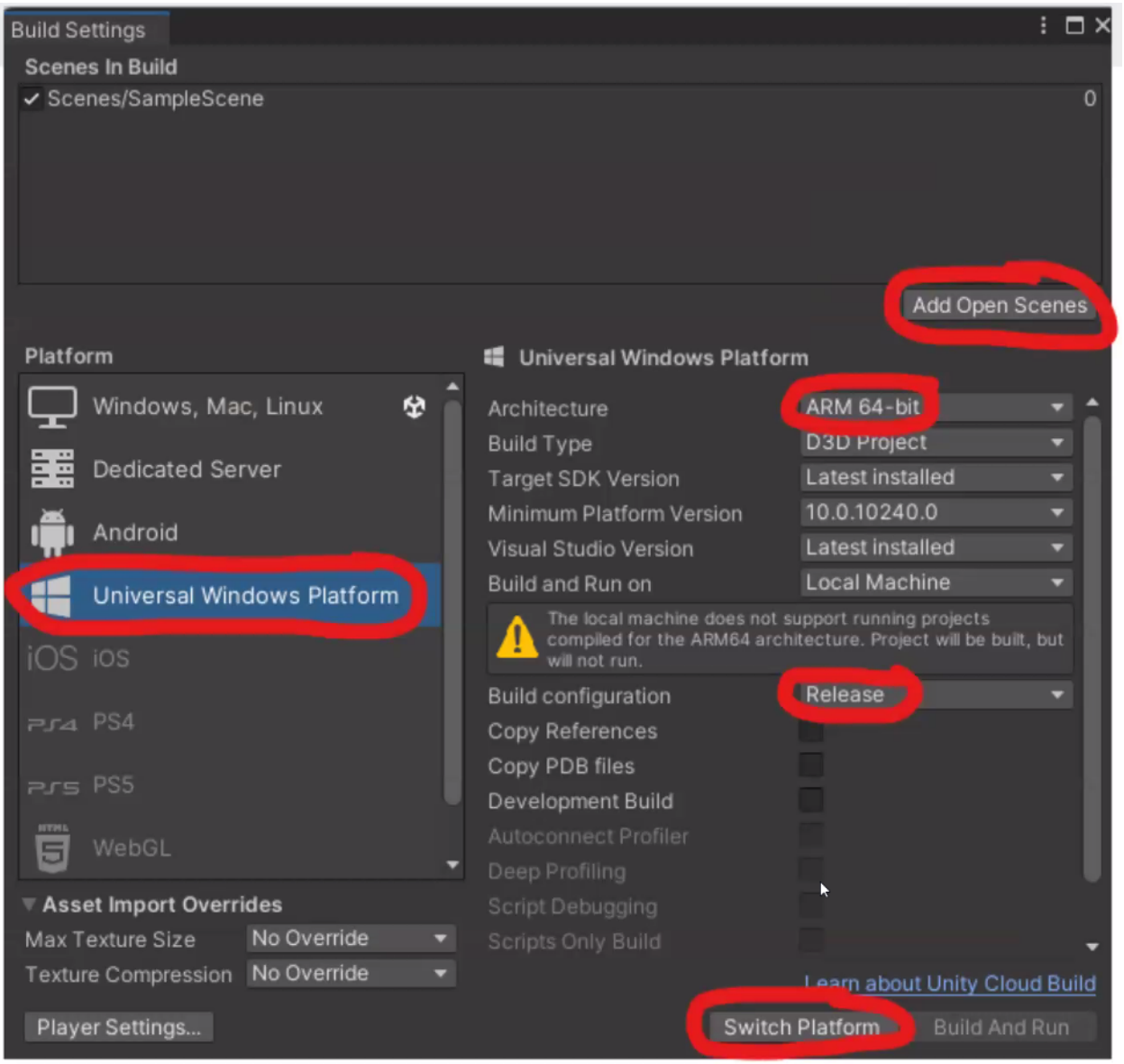


Research log

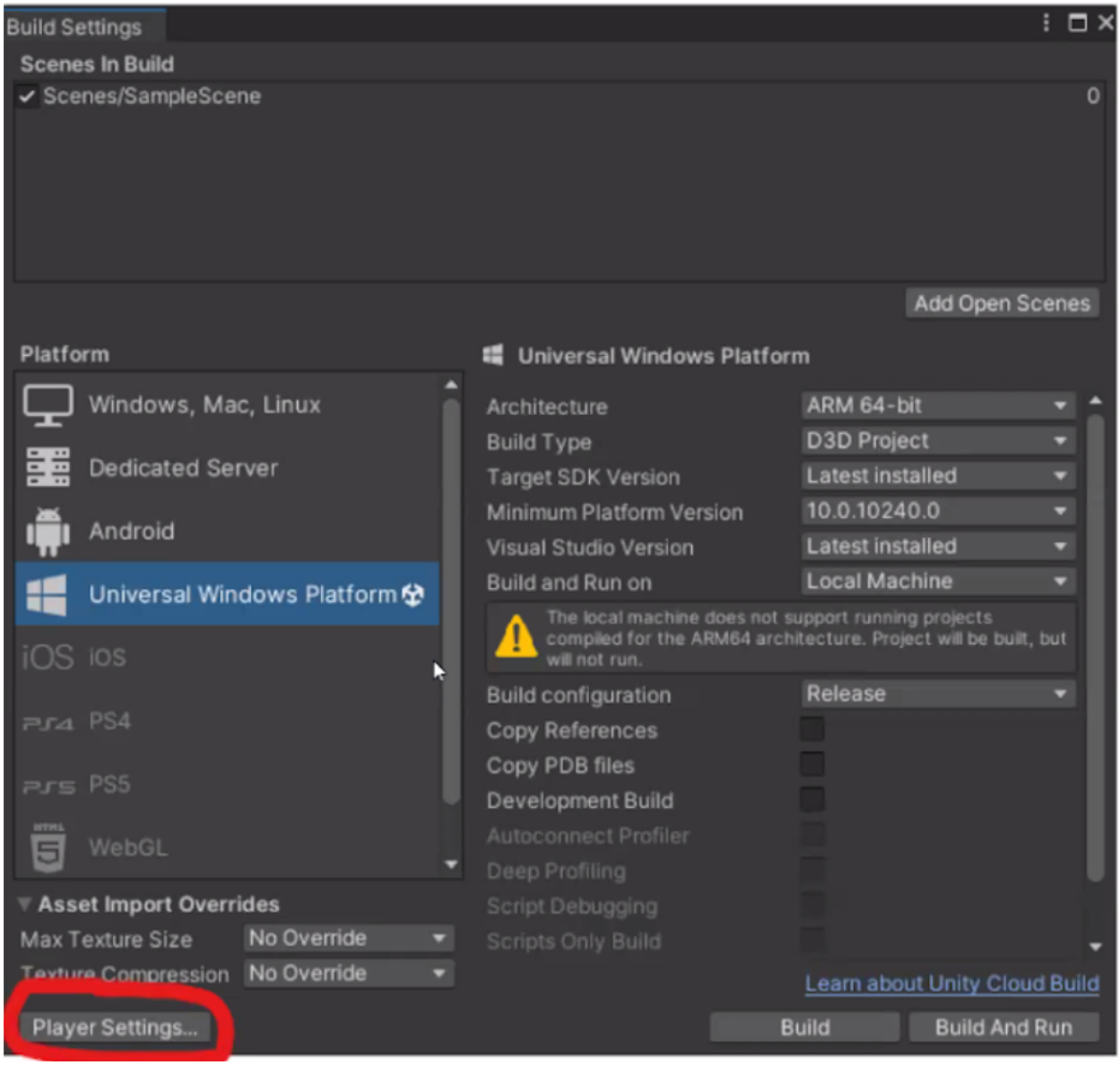
Changing Unity settings:

Build settings:

Switch the platform to that required for Hololens

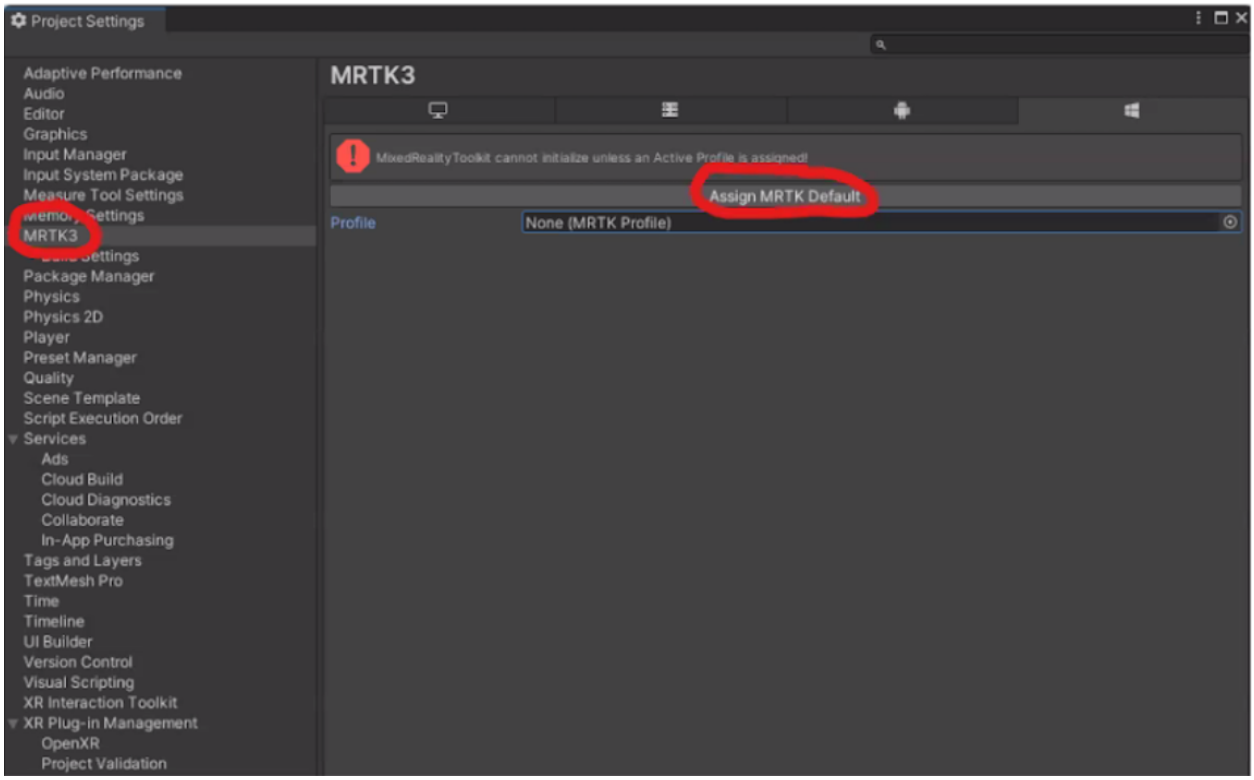


Research log

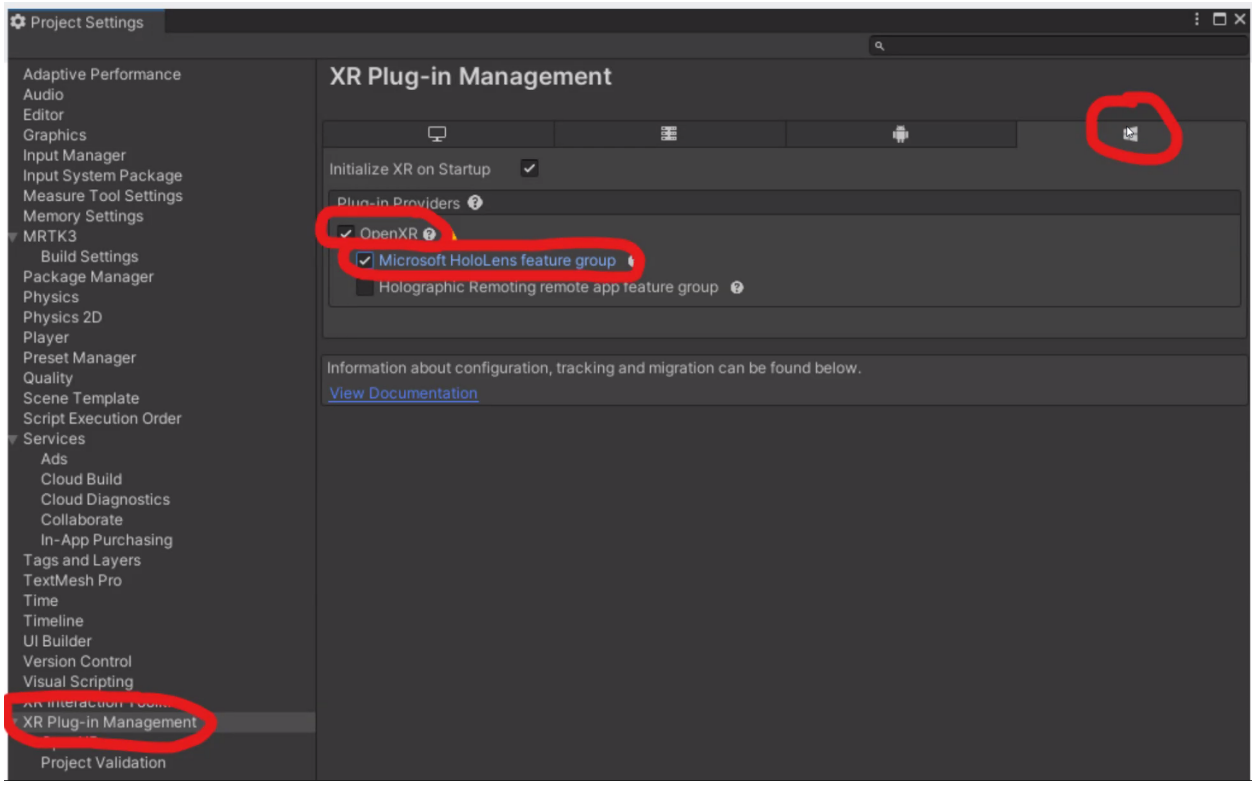


Change Project Settings:
Assign the MRTK3 default Profile

Research log

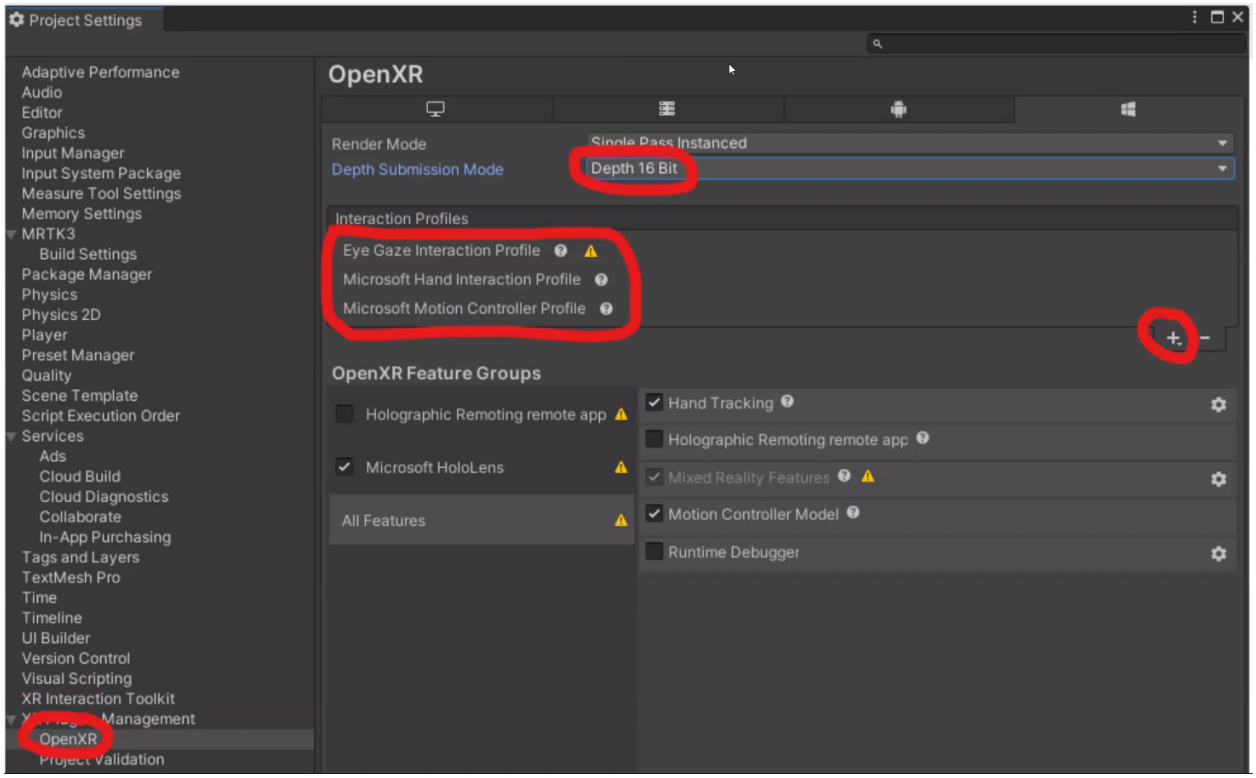


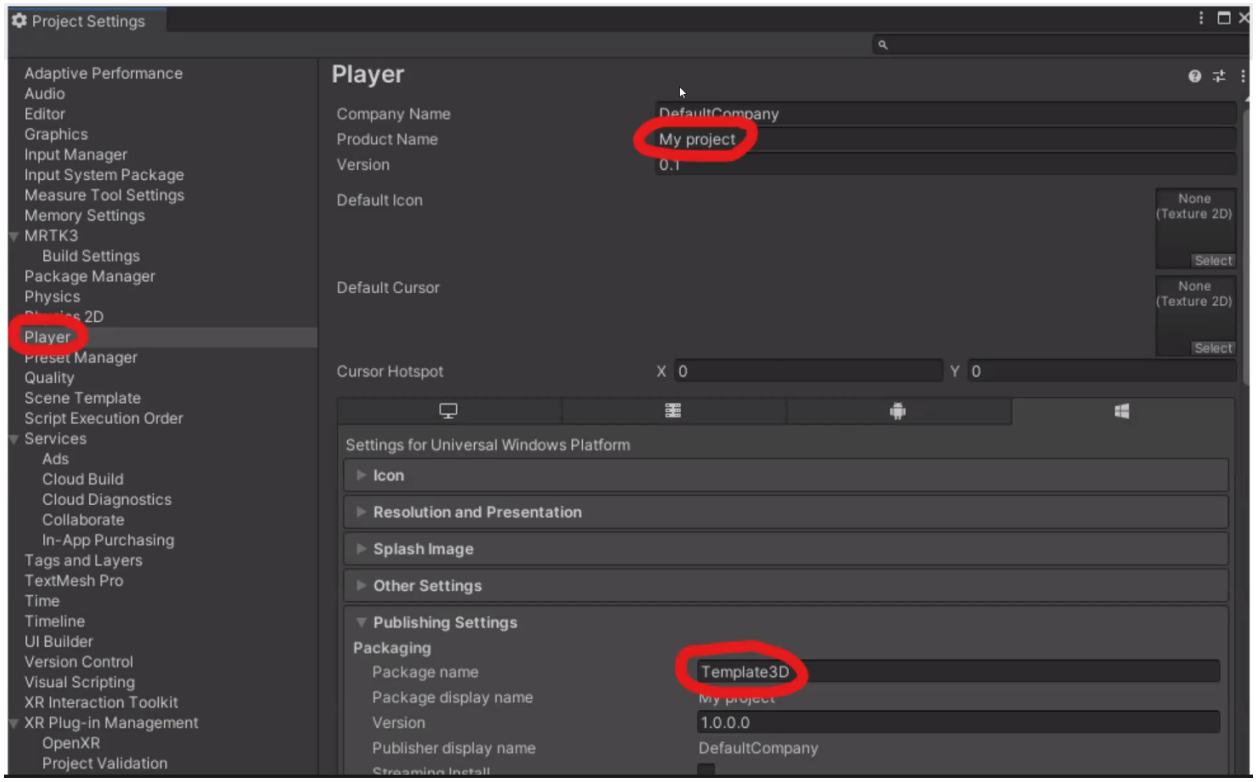
Change the XR plug in for the windows platforms



Research log

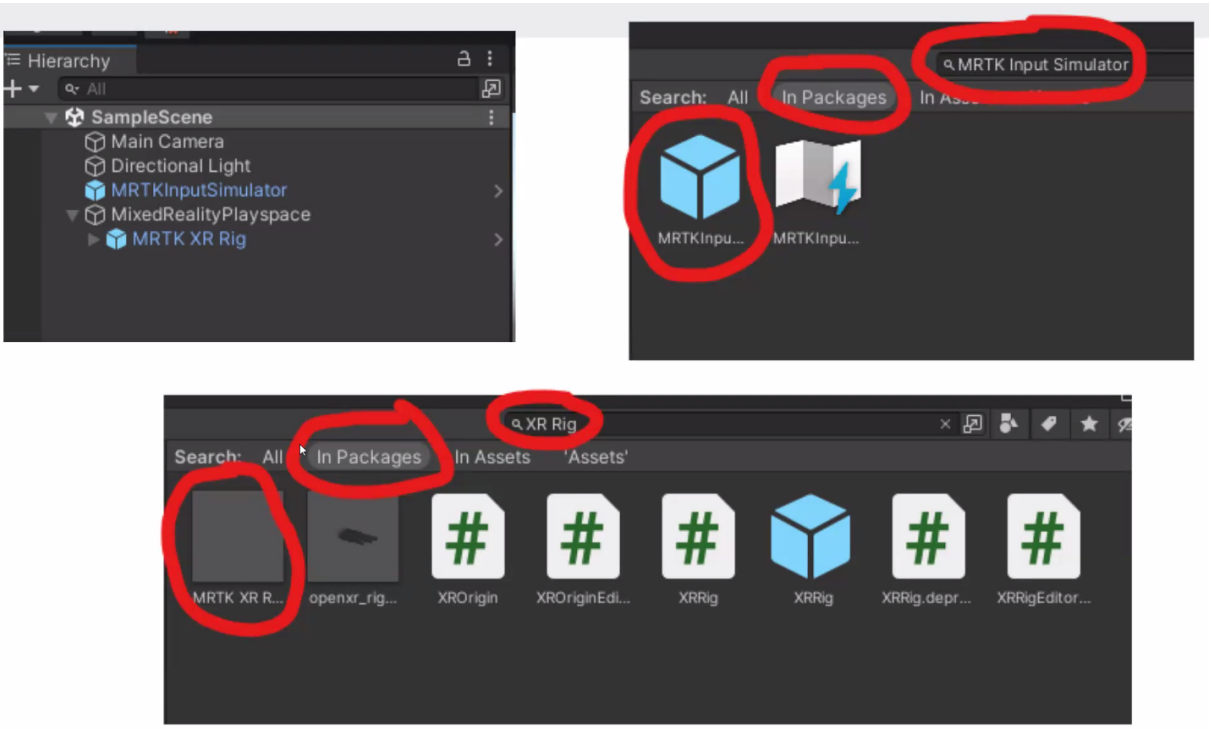
Enable the interaction profiles needed in the openXR settings



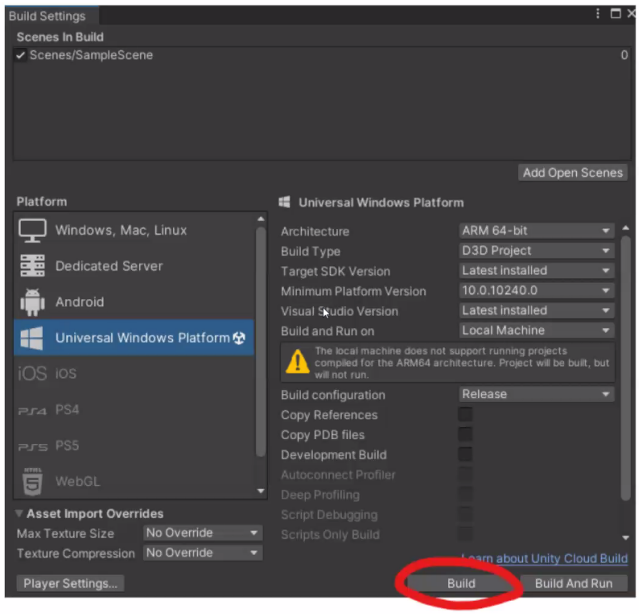


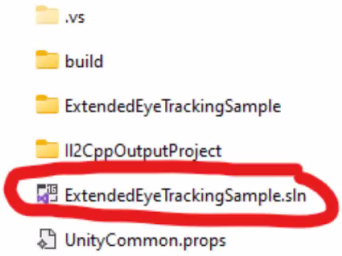
Research log

For the scene setup:
Drag the MRTKInput XR Rig and delete the default camera



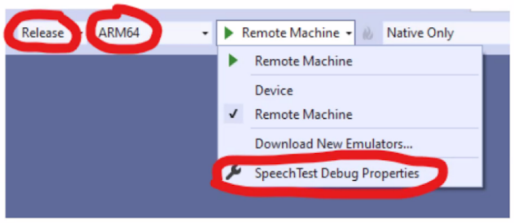
To build the application to Hololens:
Build the application from unity to get a solution file

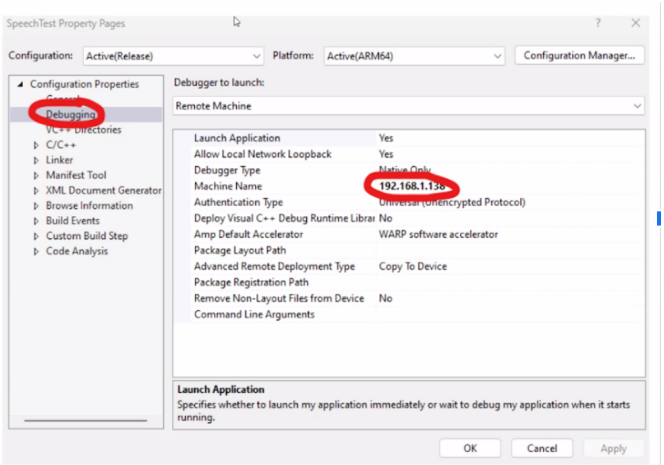




Research log

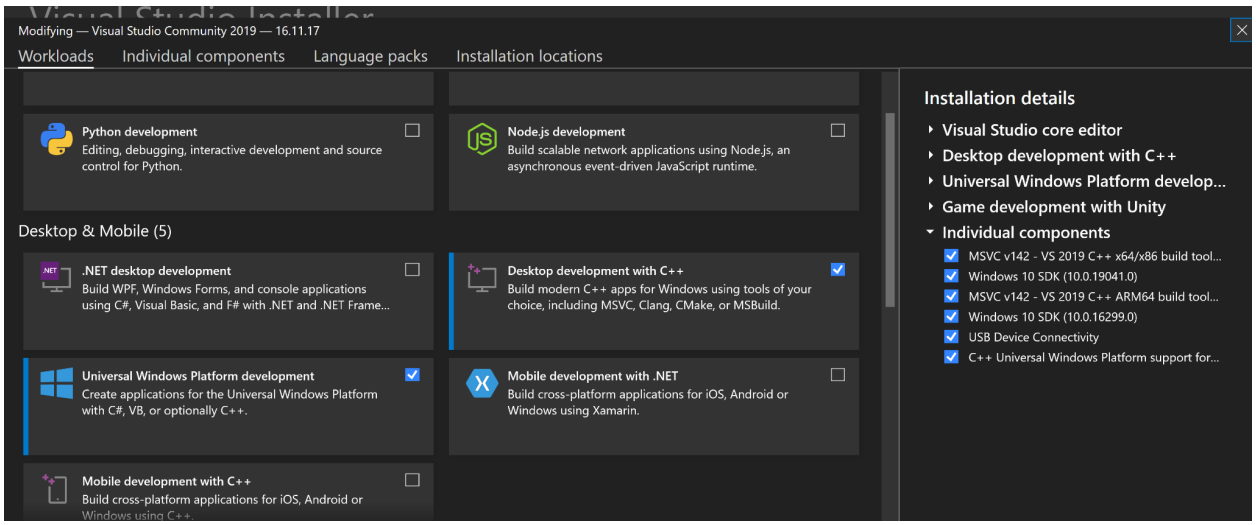
In the Visual studio



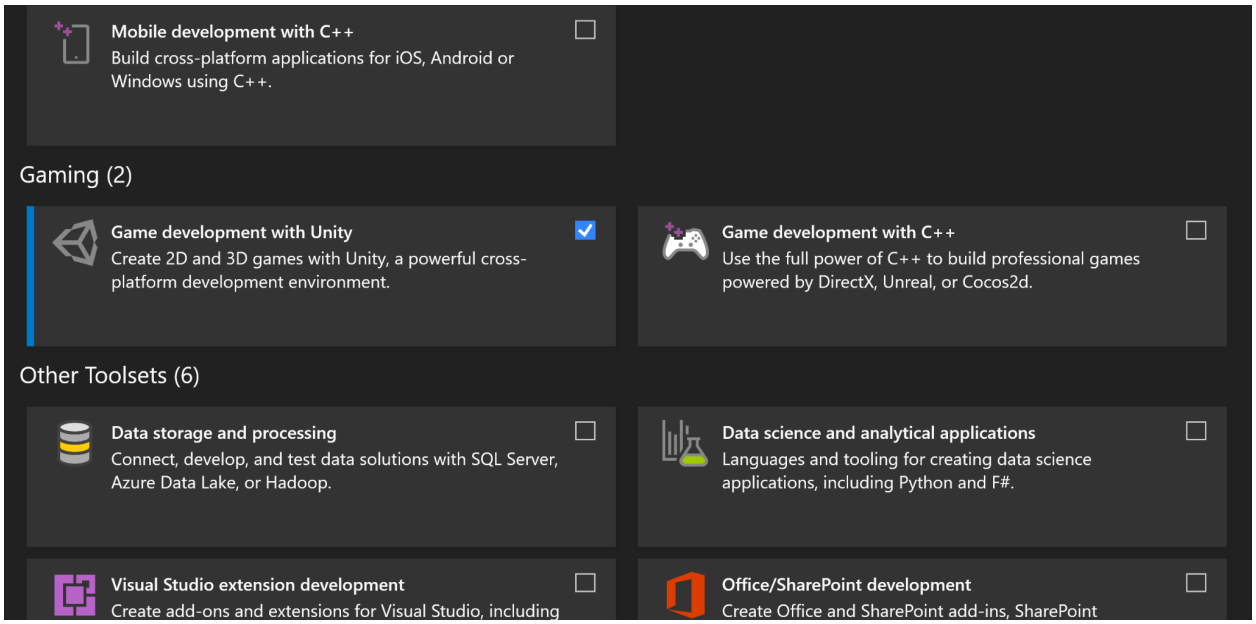


You can find the IP address of the hololens in the Network and security settings

In the visual studio installer make sure to check the desktop development with C++

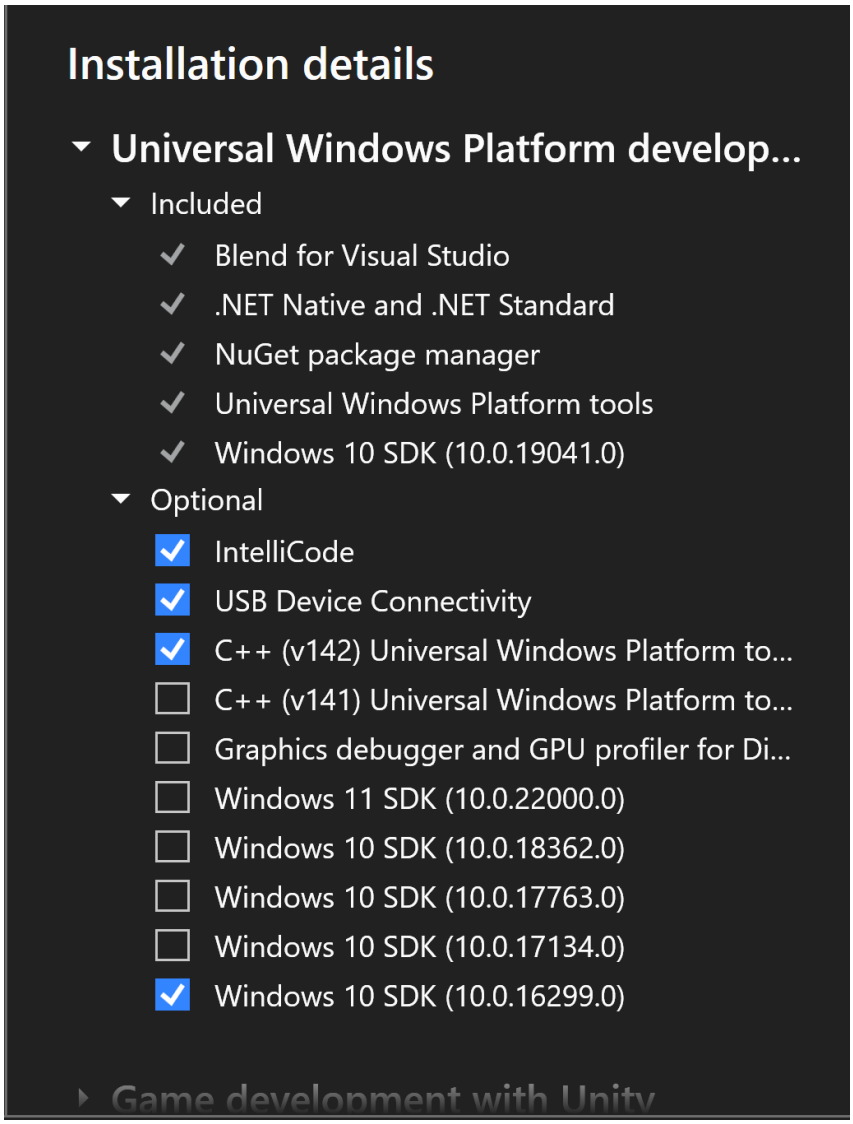


Research log



Research log

Found the issue while setting up app in the hololens, for solution tick the individual components as shown Under Universal Windows Platform Development tab.



Enable the Developer mode in the Hololens

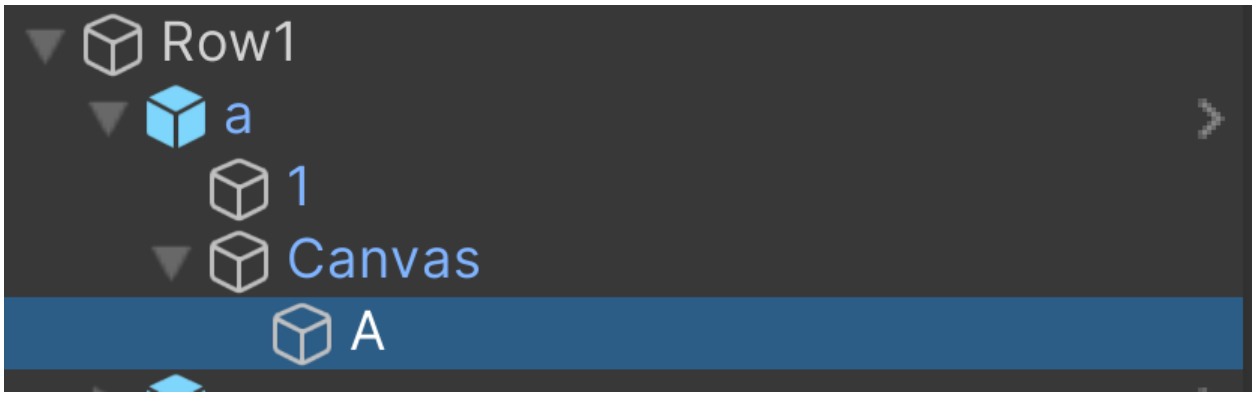
1. **Select the Update menu item.**
2. **Select the For developers menu item.**
3. **Enable Use developer features to deploy apps from Visual Studio to your HoloLens**

Research log

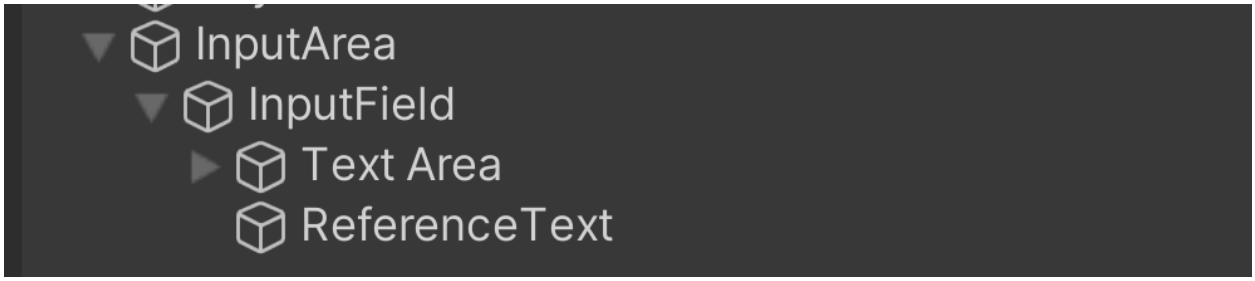
Development in Unity:

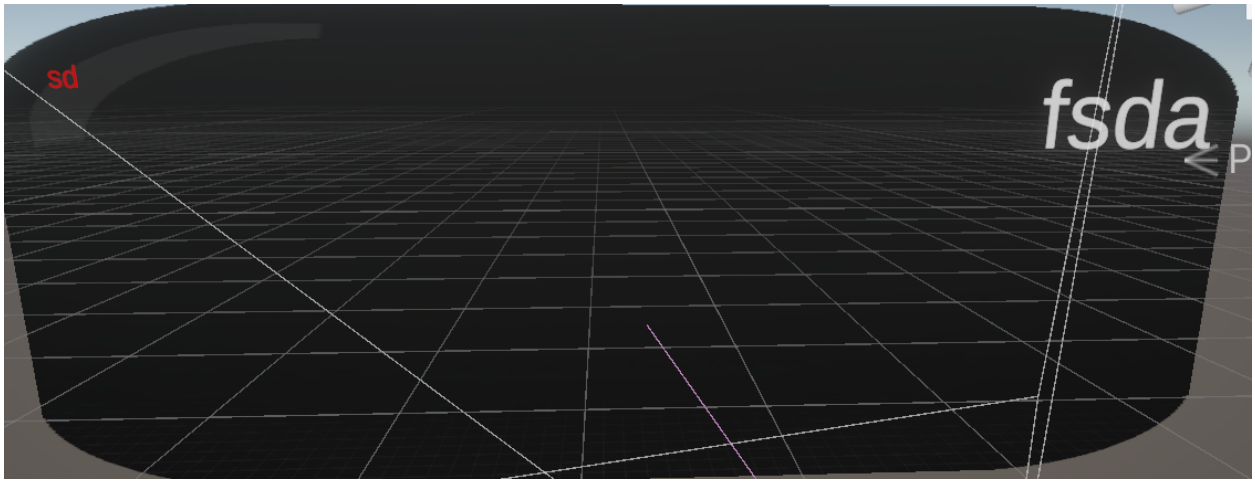
1. **Make Keys For the flower text entry layout**
   Make parent and child game objects such that it supports space key, capital letters and number entries

   Make canvas to represent the key names:



2. **Make input text area layout:**
   Make input text area to show the typed keys to the user and also give a reference text to follow.





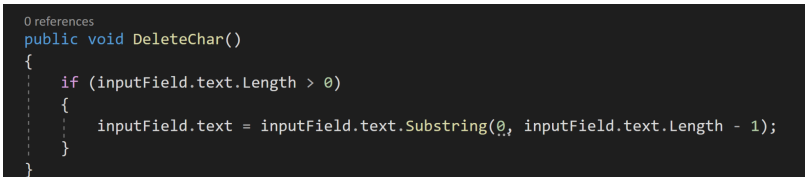
Research log

3. **Map Parent game object Key to Keys script:**
   Keys script handles all the input text entry

   Methods:
   - InsertChar

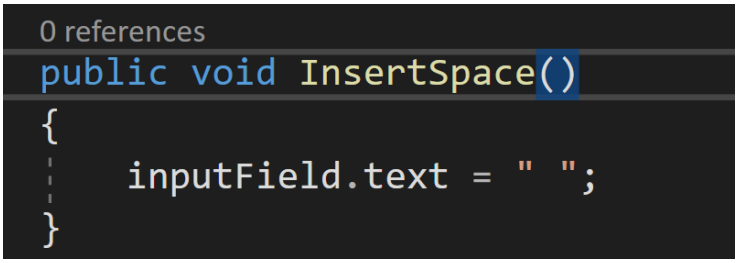
```
1 reference
public void InsertChar(string c)
{
    //int l = inputField.text.Length;
    if (a <= reference.Length)
    {
        Debug.Log("number is " + a);
    char x = reference[a];
    char z = c[0];

        if (z == x)
        {
            inputField.text += "<color=green>" + z + "</color>";



            //  inputField.image.color = Color.red;
        }
        else
        {
            inputField.text += "<color=red>" + c + "</color>";
        }
        a++;
    }
}
```

   handles the insert character. Checks if the character matches the character int the reference string and changes the color of text to red if not matched and green if matched
   - DeleteChar



```
0 references
public void DeleteChar()
{
    if (inputField.text.Length > 0)
    {
        inputField.text = inputField.text.Substring(0, inputField.text.Length - 1);
    }
}
```

   for the backspace key press
   - InsertSpace
     Handles the space bar key press



```
0 references
public void InsertSpace()
{
    inputField.text = " ";
}
```

   - CapsPressed
     Handles the Caps lock

Research log

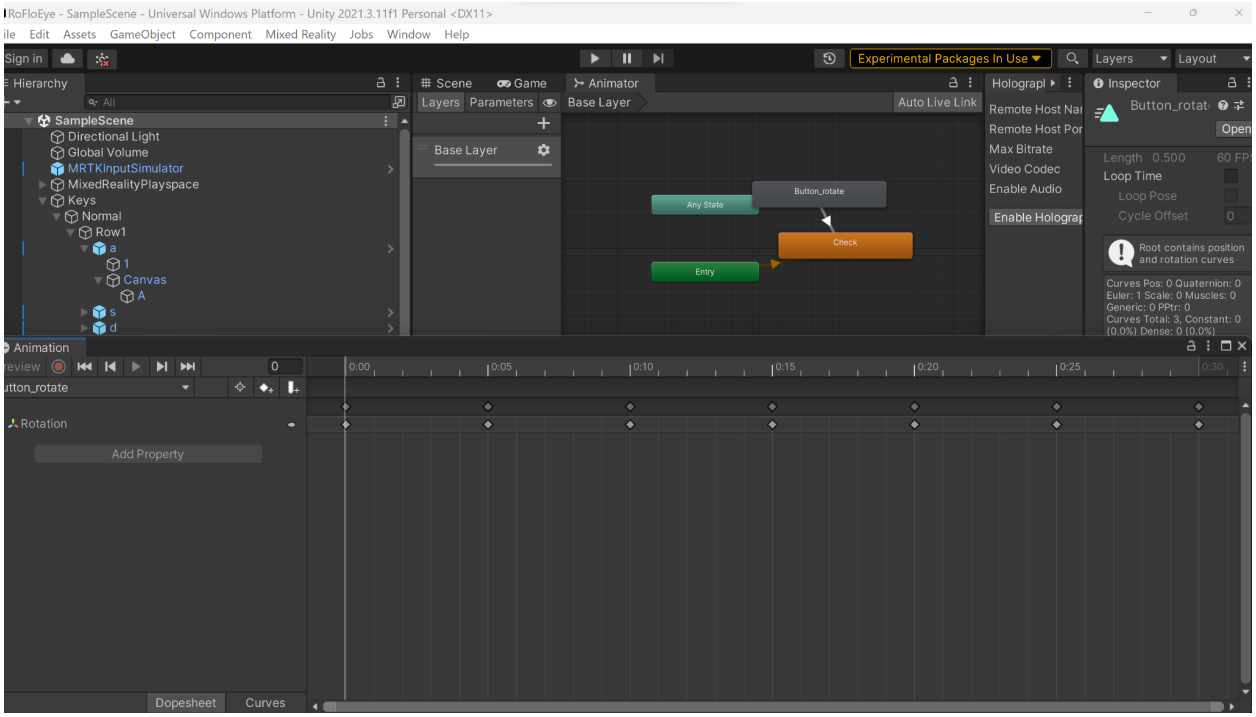
```
public void CapsPressed()
{
    if (caps)
    {
        normalButtons.SetActive(false);
        capsButtons.SetActive(true);
        caps = true;
    }
    else
    {
        capsButtons.SetActive(false);
        normalButtons.SetActive(true);
        caps = false;
    }
}
```

4. **Map each key game object to KeyboardButton Scrept:**
   This script handles the input triggers and onclicks.

```
Assembly-CSharp | KeyboardButton | OnContinousStare()
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using TMPro;
5   using UnityEngine.UI;
6
    Unity Script (1 asset reference) | 0 references
7   public class KeyboardButton : MonoBehaviour
8   {
9       Keys keys;
10      Collider m_Collider;
11      TextMeshProUGUI buttonText;
12      public GameObject button;
13      public Material Material1;
14      [SerializeField] float _degreesPerSecond = 30f;
15      [SerializeField] Vector3 _axis = Vector3.forward;
16
17      // Start is called before the first frame update
        Unity Message | 0 references
18      void Start()
19      {
20          keys = GetComponentInParent<Keys>();
21          buttonText = GetComponentInChildren<TextMeshProUGUI>();
22          if(buttonText.text.Length == 1)
23          {
24              NameToButtonText();
25          }
26          m_Collider = GetComponent<Collider>();
27      }
28
```

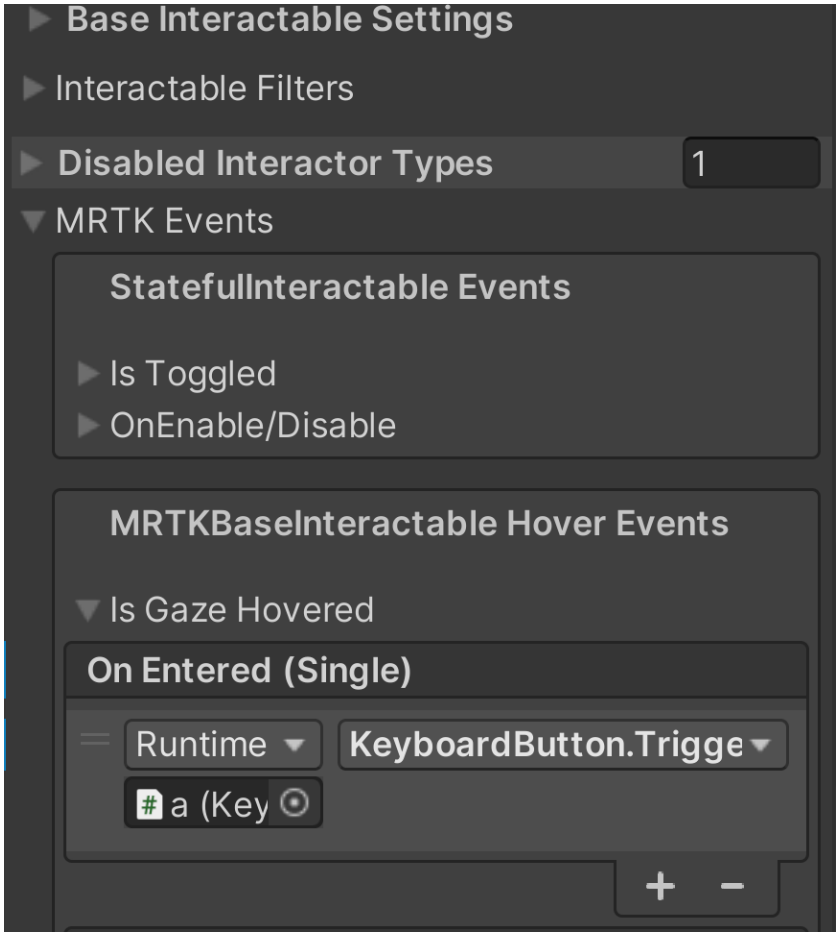
5. **Make Animation that can be called when key is pressed:**

Research log



6. **Add Object manipulator component to each key and map the Keyboard button scripts to the OnEnter component in the is gaze hovered:**

Research log



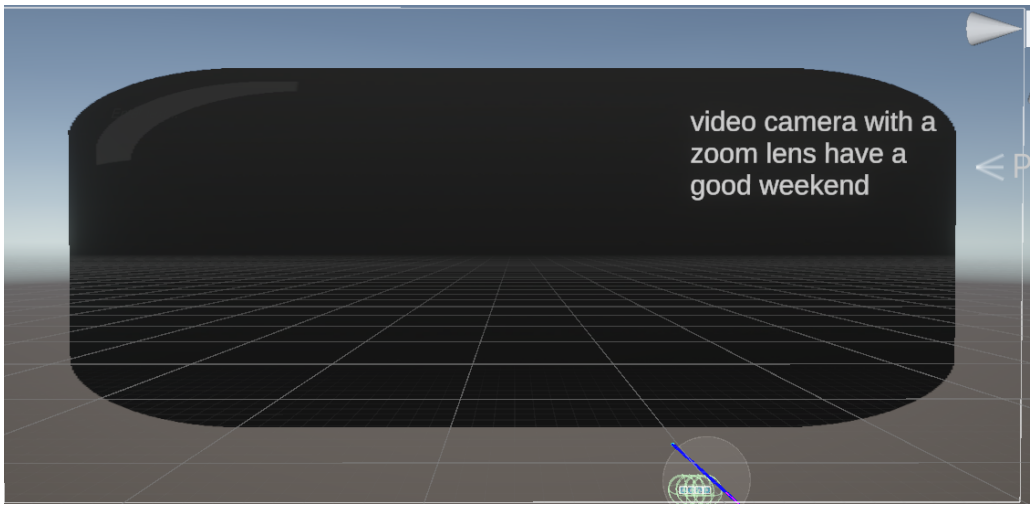
Research log

7. **Change the color of button and start the animation once the gaze hovered is confirmed**

```
0 references
public void Button_click_register()
{
    keys.InsertChar(buttonText.text);
    m_Collider.enabled = false;
    GetComponent<Animator>().Play("Button_rotate");
    m_Collider.enabled = true;
}

1 reference
public void NameToButtonText()
{
    buttonText.text = gameObject.name;
}

0 references
public void Triggered()
{

    button.GetComponent<SpriteRenderer>().material = Material1;

}
```

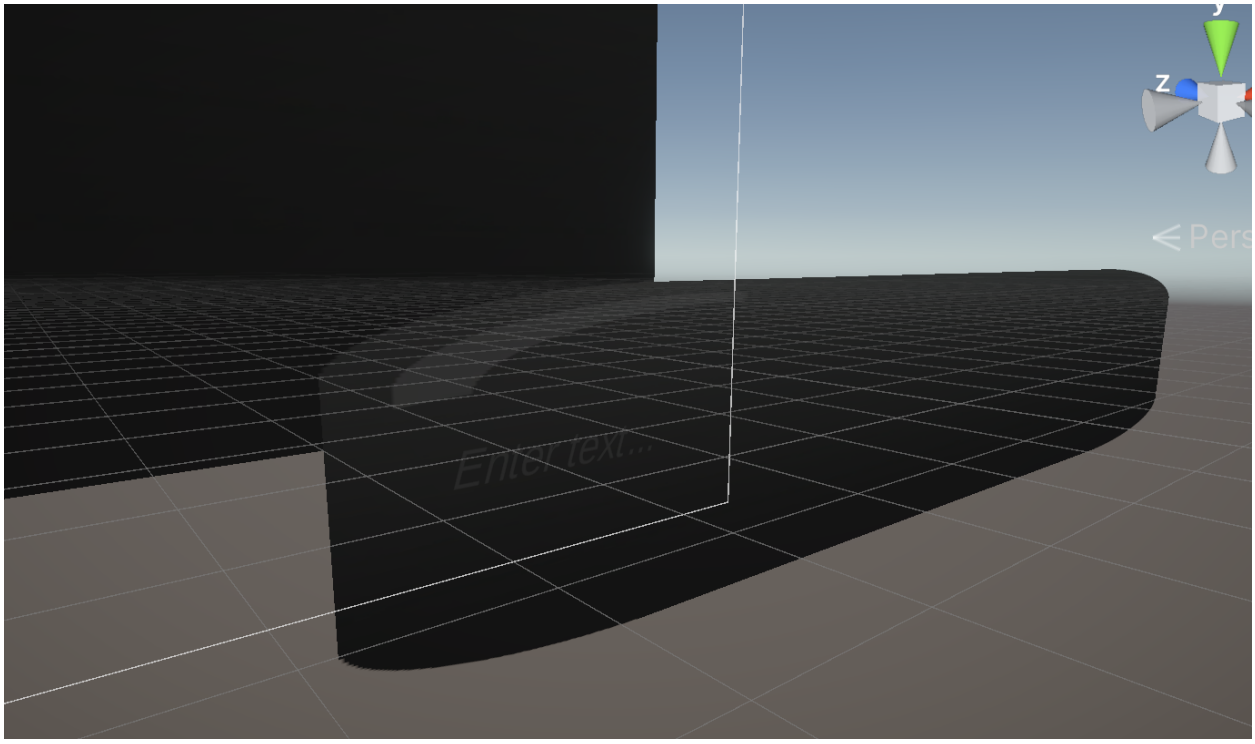
8. **Change the Animation to rotation on triggered and move the animation to OnClickRegister**

9. **Change the reference text to Phrase sets for evaluating text entry techniques given by I. Scott MacKenzie**



Research log

10. **Make another input Text area for words typed and error display.**



11. **Write code to get the number of words typed and error, i.e introduce variables and increment them in code.**

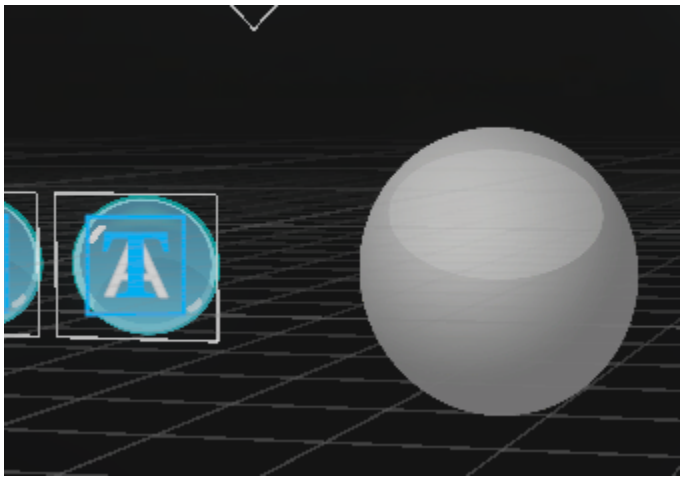
```
        if (z == x)
        {
            inputField.text += "<color=green>" + z + "</color>";



            //  inputField.image.color = Color.red;
        }
        else
        {
            inputField.text += "<color=red>" + c + "</color>";
            error++;
        }
        a++;
    }
    countInput.text = "Word Count : " + wordcnt + ", " + "<color=red>" + "Error : " + error + "</col
//}
```

12. **Put dwell time in the code for each key**

Research log

```
Unity Message | 0 references
private void Update()
{
    if (select_flag)
    {
        timer -= Time.deltaTime;

        canvas.transform.Rotate(0, 0, 1 * _degreesPerSecond * Time.deltaTime);
       // Debug.Log("outside " + canvas.transform.eulerAngles.z);
        if (timer <= 0.0f)
        {
            Debug.Log("in rthe if");
            Button_click_register();

        }
    }
}
```

13. Create a Center grey button

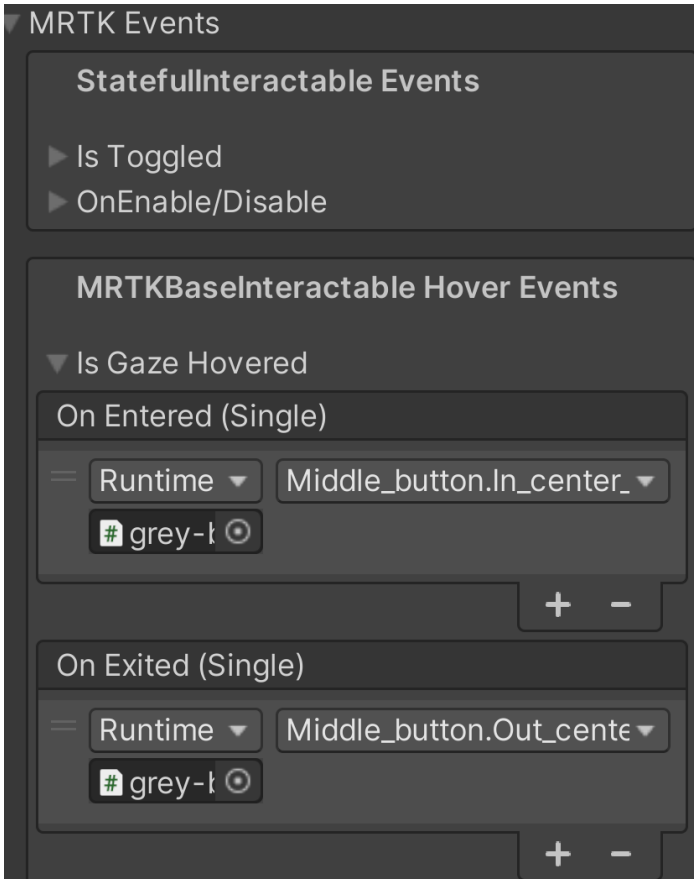


Research log

14. **Create the Center button script that has methods to change the material when gaze ray hovers on it**

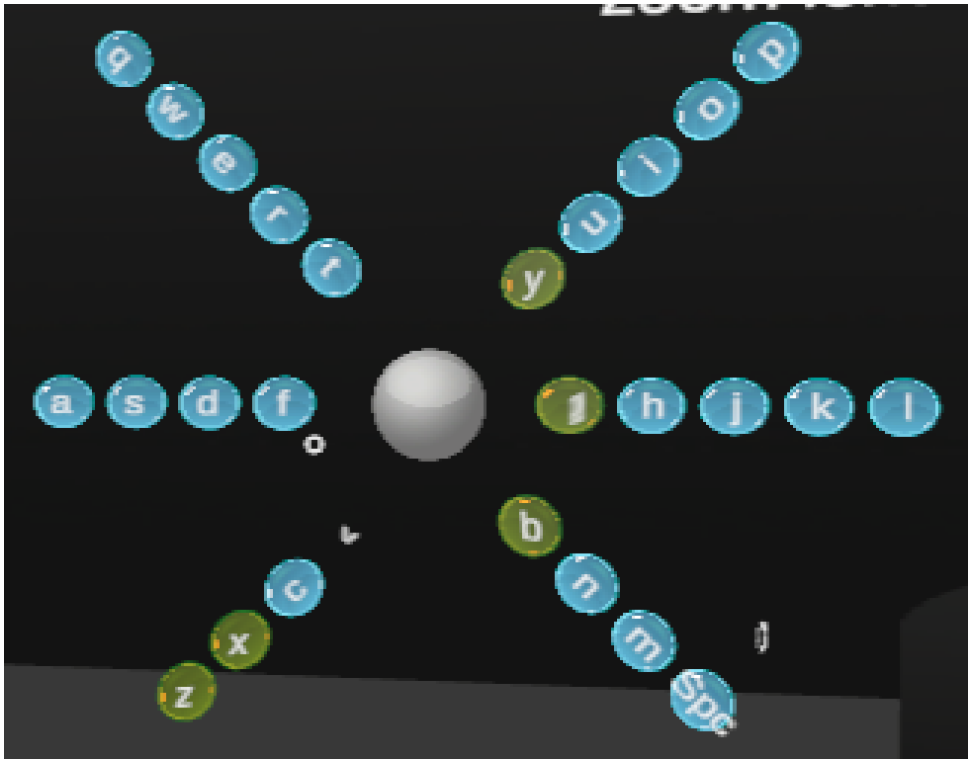
```
Unity Message | 0 references
void Update()
{

}
0 references
public void In_center_button()
{
    Debug.Log("in the collider");
    button.GetComponent<SpriteRenderer>().material = In_center;
}

0 references
public void Out_center_button()
{
    button.GetComponent<SpriteRenderer>().material = Out_center;
}
}
```

15. **Map the script to the on-enter and on-exit function calls in the MRTK events of the center button**

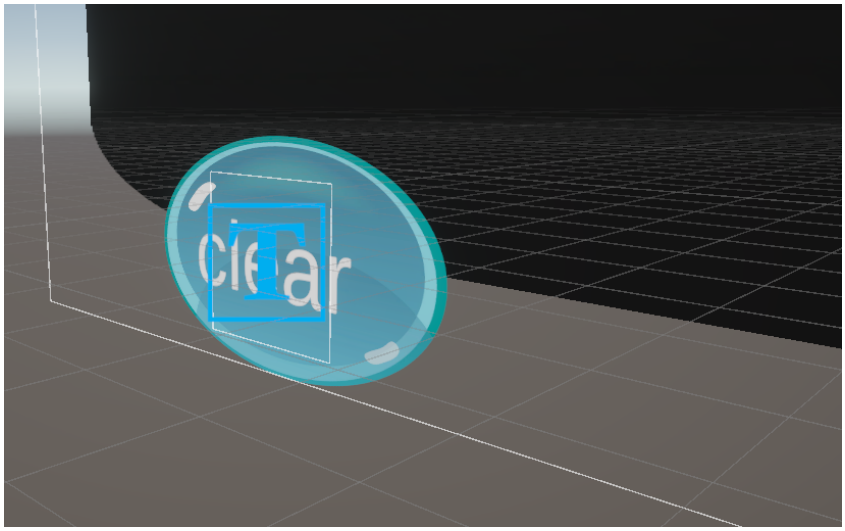


Research log

16. Create the other buttons to complete the layout of the flower Text entry



17. Create a clear button to allow the user to restart the study if need be



18. Create a program to enable the clear button functionality

```
1 reference
public void Clear()
{
    inputField.text = "";
    a = 0;
    wordcnt = 0;
    error = 0;
    countInput.text = "Word Count : " + wordcnt + ", " + "<color=red>" + "Error : " + error + "</color>";
}
```

Research log

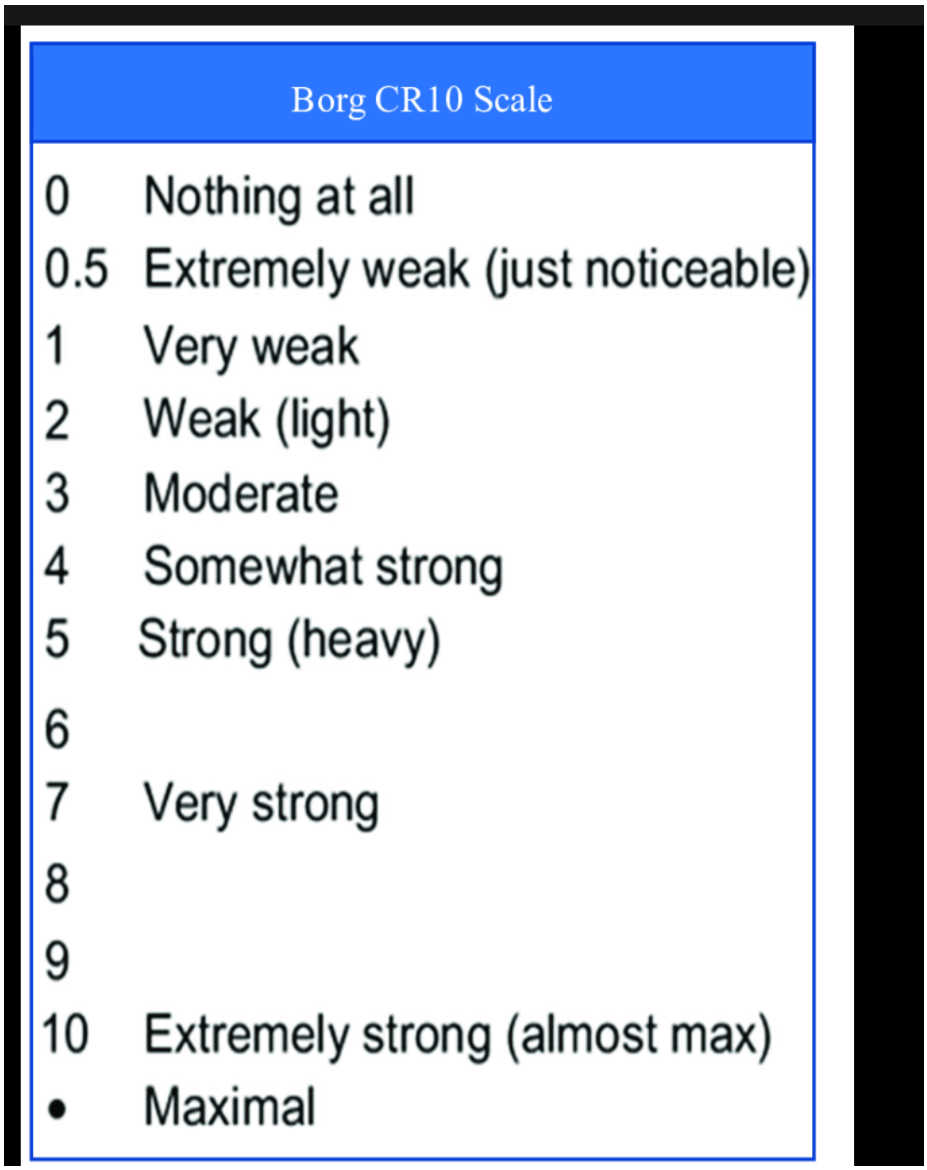
19. Map the Clear function to the KeyboardButton script

```
ublic void Button_click_register()

  if (!String.Equals(gameObject.name.ToLower(), "Spc".ToLower()) && !String.Equals(gameObject.name.ToLower(),"clea
      keys.InsertChar(buttonText.text);
  else
      if (String.Equals(gameObject.name.ToLower(), "clear".ToLower()))
  {
      keys.Clear();
  }
  else
      keys.InsertSpace();
     button.GetComponent<SpriteRenderer>().material = SelectMaterial;
      //m_Collider.enabled = false;
      GetComponent<Animator>().Play("Button_rotate");

  timer = 1.00f;
      //select_flag = false;
      //m_Collider.enabled = true;
```

Research log

User Study:

- Use the Borgs scale for a rating on exertion and eye fatigue.



- Interview questions to understand the demographic including familiarity or previous usage of Augmented reality devices and eye-tracking applications.
- Making the user instructions and ideas on getting the user familiar with the interface and the device i.e Hololens 2.
- Informing users of the timer and the goal to achieve.
- Analyzing the data from the application i.e the words typed and the error to get the error rate and the words per minute.
- Analyzing the data from the interview and ratings from the user.

Ex User data

Dwell times	Words Per minute	Error rate

Research log

4 seconds	3.5	0
1 second	8	2
0.5 seconds	11	5

Interaction variations

1. Layouts and interface methods tested.
    a. Rotating Flower text entry.
    b. Standard Gaze typed keyboard.
    c. Air typing keyboard of hololens.
2. Different Dwell times tried on the Rotating flower text entry layout.
    a. 4 seconds (for trial and familiarizing the interface to user)
    b. 1 second (hypothesized to be the best dwell time for regular users)
    c. 0.5 seconds (hypothesized to be done well by advanced users)