WHMIS PART 2 : Final Draft

- part 1 documented some of the development process and earlier drafts of the code
- part 2 documents the final codes with all changes included.

Universal Functions Script

- holds most of code with some references to smaller scripts

① check the beginning of the ~~Universal functions Script~~ to see all variable declarations

Station 1

- This section will detail all code pertaining to STN 1

1) update function in universal functions script

```
// RESTRICTIONS FOR STATION 1, ONCE PLAY IS PRESSED, RESTRICTIONS ARE ACTIVATED UNTIL ALL CORRECT MATCHES ARE MADE----------
if (startGame) // when at the start button at the beginning of the game is pressed, this is enabled to start the restrictions
{
        mainCanvasText.text = "here are your scores so far\n" +
                              $"Station 1: You have correctly guessed {pointCounterStation1Script.points} labels, you have had {pointCounterStation1Script.wrong}  wrong attempts so
    if (currentPosition.x < minXLimit)
    {
        currentPosition.x = minXLimit;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.x > maxXLimit)
    {
        currentPosition.x = maxXLimit;
        mainCamera.transform.position = currentPosition;
    }

    if (currentPosition.z > maxZLimit)
    {
        currentPosition.z = maxZLimit;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.z < minZLimit)
    {
        currentPosition.z = minZLimit;
        mainCamera.transform.position = currentPosition;
    }
}
```

↳ updates score display on hand menu as game is played

Restricts movement on x and z axes while game is in play

FALSE AND STATION 2 HAS NOT BEEN COMPLETED (UNRESTRICT1 = FALSE)----------

function called when start button is pressed:

```
public void Station1Restrict()
{
    //called when play button is pressed to activate first set of restrictions
    startGame = true;
}
```

2) each Bin has it's own script attached that are nearly identical .. here is an example:

↓

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


Unity Script (2 asset references) | 0 references
public class BioHazardBin : MonoBehaviour
{
    public bool hasCollidedBioHazard = false; // Flag to track if collision has occurred
    public UniversalFunctions universalFunctionsScript; // Reference to universal functions script
    public GameObject cube;
    public GameObject label;
    public Vector3 positionEnd = new Vector3(9.6f, 0.8f, 9.5f);
    public Vector3 rotationEnd = new Vector3(0, 0, 0);
    public Vector3 beginningPosition;
    public GameObject spawnManager1;

    Unity Message | 0 references
    private void Start()
    {
        beginningPosition = spawnManager1.transform.position; // on start get the position of the spawn manager.
    }
    Unity Message | 0 references
    private void OnCollisionEnter(Collision collision) // When a collision occurs between any object with a collider, and this object that the script is attached to.
    {

        if (collision.gameObject.CompareTag("BiohazardLabel") && !hasCollidedBioHazard)
        {   if the collision occurs with an object tagged  and the collision has not occured before
            hasCollidedBioHazard = true; // Mark collision as occurred
            ~~Debug.Log("collide");~~
            PointCounterStation1.Instance.AddPoints(1); // Adds 10 points
            universalFunctionsScript.ChangeMaterial(Color.green, cube); // changes colour of cube

            Debug.Log("Moving label from " + label.transform.position + " to " + positionEnd);
            label.transform.position = positionEnd; // Puts label in certain position and rotation
            label.transform.rotation = Quaternion.Euler(rotationEnd);

            Rigidbody rb = label.GetComponent<Rigidbody>(); // gets the rigidbody component of the label
            if (rb != null)
            {
                Debug.Log("freeze all");
                rb.constraints = RigidbodyConstraints.FreezeAll; // Freeze position and rotation
            }
        }
        else if (!collision.gameObject.CompareTag("BiohazardLabel") && !hasCollidedBioHazard) // Clause for collision with wrong labels.
        {
            Debug.Log("Collision with other object");
            universalFunctionsScript.WrongGuess();
            universalFunctionsScript.ChangeMaterial(Color.red, cube); // Change cube color to red for other collisions
            //Move the collided object to beginningPosition
            Debug.Log(beginningPosition);
            Rigidbody rb = collision.gameObject.GetComponent<Rigidbody>();
            if (rb != null)
            {
                // Move the collided object to beginningPosition
                rb.MovePosition(beginningPosition);
            }
        }
    }

}
```

5) point counter stn 1 script has all the functions for wrong guess and add points mentioned in the script.

```
// Method to add points
9 references
public void AddPoints(int amount)
{
    points += amount;
    UpdatePointText();

    if (points == 9) // when all labels have been correctly matched
    {
        universalScript.GameOver1();
    }
}
1 reference
public void WrongGuess(int amount)
{
    wrong += amount;
    UpdatePointText();
}
```

adds wrong guess and updates text.

4) The ChangeMaterial function in the Universal Functions Script

```
// Change material color of an object
18 references
public void ChangeMaterial(Color color, GameObject cube)
{
    Renderer otherRenderer = cube.GetComponent<Renderer>();
    if (otherRenderer != null)
    {
        otherRenderer.material.color = color;
    }
}
```

takes in a Game Object, does not have to be a cube, I just used that as a variable name.

takes in a colour

get the renderer component of the game object

5) game over 1 function from the Universal Functions Script

```
1 reference
public void GameOver1()
{
    ~~Debug.Log("functn gmovr1 called");~~
    startGame = false; // removes restrictions

    //updates text
    mainCanvasText.text = "Congratulations you have completed station 1, please progress to station 2!\n" +
                          "here are your scores so far\n" +
                          $"Station 1: You correctly guessed 9/9 labels, you had {pointCounterStation1Script.wrong}  wrong attempts";

    //shows the end canvas to move on to next station
    EnableCanvas(Station1End);
    game1played = true; // flags that game 1 has been played all the way through

}
```

universal functions Script

```
539     public void ClearCanvas(Canvas canvasToHide)
540     {       // retrieve all text components that are children of the specified canvas.
541         Text[] texts = canvasToHide.GetComponentsInChildren<Text>();
542         foreach (Text word in texts)
543         {
544             word.enabled = false;  // makes each text invisible
545         }
546     }
547
548     // Hide the canvas from the screen
        13 references
549     public void HideCanvas(Canvas canvasToHide)
550     {
551         canvasToHide.enabled = false;  // hides the canvas and all UI elements it has
552     }
553
554     // Enable the canvas on the screen
        6 references
555     public void EnableCanvas(Canvas canvasToEnable)
556     {
557         canvasToEnable.enabled = true;  // sets "enabeled" property of canvas to true
558         Text[] texts = canvasToEnable.GetComponentsInChildren<Text>();
559         foreach (Text word in texts)
560         {
561             word.enabled = true;
562         }
563     }
```

makes sure each text component is also enabled

```
565     // Toggle the canvas visibility with the controller
        1 reference
566     void ToggleCanvasVisibility(Canvas canvas)
567     {
568         // Toggle enabled component of canvas, if canvas is currently enabled, it will be disabled and the opposite.
569         canvas.enabled = !canvas.enabled;
570         Text[] texts = canvas.GetComponentsInChildren<Text>();
571         foreach (Text text in texts)
572         {
573             text.enabled = canvas.enabled;
574         }
575
576
577     }
```

shows as a boolean (ie. true or false)

set enabeled status of text to match that of canvas

Station 2

- This section will detail all code pertaining to STN 2

1) restriction function in universal functions script

```
public void Station2Restrict()
{
    //called when button clicked at end of station1
    mainCamera.transform.position = new Vector3(-40, 0, 15); // transports camera to next station

    mainCamTransform.localRotation = Quaternion.identity; // Reset local rotation
    Quaternion newRotation = Quaternion.Euler(0f, -90f, 0f); // ~~Example Quaternion~~ Sets to a new rotation to face start of station
    mainCamTransform.rotation = newRotation;

    gamelover = true; // starts station 2 restrictions when button is pressed

    stn2Effect.Play(); // plays effect on floor
    // Start coroutine to stop the particle system after 10 seconds
    if (stopParticleCoroutine != null)
    {
        StopCoroutine(stopParticleCoroutine);
    }
    stopParticleCoroutine = StartCoroutine(StopParticleSystemAfterDelay(10f, stn2Effect));
}
```

2) Coroutine function in Universal Functions Script

takes in a float for the delay

takes in something of type particle system

```
4 references
private IEnumerator StopParticleSystemAfterDelay(float delay, ParticleSystem particle)
{
    //coroutine for particle systems
    // Wait until the specified delay is over
    yield return new WaitForSeconds(delay);
    // Stop the particle system
    if (particle != null)
    {
        particle.Stop();
    }
}
```

3) Update function in the universal functions Script (only the section of the update fcn that pertains to STN 2)

```
//RESTRICTIONS FOR STATION 2, ONLY TRUE WHEN STARTGAME IS FALSE AND STATION 2 HAS NOT BEEN COMPLETED (UNRESTRICT1 = FALSE)--

if (!unrestrict1 && !startGame && gamelover)
//restrictions are on (ie. you have not completed the station, station1 is false (ie. game has been played),
//and you have pressed the button on the panel at the end of game1
{
    Debug.Log("stn2 restricts");
    if (currentPosition.x < minXLimit1)
    {
        currentPosition.x = minXLimit1;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.x > maxXLimit1)
    {
        Debug.Log("max lim change pos");
        currentPosition.x = maxXLimit1;
        mainCamera.transform.position = currentPosition;
    }

    if (currentPosition.z > maxZLimit1)
    {
        currentPosition.z = maxZLimit1;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.z < minZLimit1)
    {
        Debug.Log("min lim change");
        currentPosition.z = minZLimit1;
        mainCamera.transform.position = currentPosition;
    }
}
```

used to display statements in the Console during runtime. Not relevant to code functionality

4) Script attached to each socket interactor (example of the one for the Harmful Hazard)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

Unity Script (3 asset references) | 0 references
public class HarmfulSocketInteractor : MonoBehaviour
{
    public UniversalFunctions universalFunctionsScript; // declares a reference to our universal functions script.

    Unity Message | 0 references
    private void OnTriggerEnter(Collider other) // on collision w/ another object
    {
        // Check if the entering object has the correct tag
        if (other.CompareTag("HarmfulHazardLabel"))
        {
            universalFunctionsScript.harmfulIn = true;
            }
        else
        {
            universalFunctionsScript.harmfulIn = false;
        }
    }
}
```

reference to script

variable in referenced script

5) Another section of the update function in the universal functions Script

```
        // Check conditions for SDS section
        if (harmfulIn && HealthIn && FlammableIn)
        {
            GameOver2(); // Calls Fcn to signal end of game 2
        }
```

each variable is not switched to true until the right Labels are put into the right zones as seen in the example in section 4.

6) Game Over 2 function in the universal functions Script

```
1 reference
public void GameOver2()
{
    //updates text
    mainCanvasText.text = "Congratulations you have completed station 1, please progress to station 2!\n" +
                          "here are your scores so far\n" +
                          $"Station 1: You correctly guessed 9/9 labels, you had {pointCounterStation1Script.wrong}  wrong attempts" +
                          "\n Station 2: you correctly matched 3/3 of the SDS labels";

    unrestrict1 = true; // stops restrictions
    EnableCanvas(stn2END); // enables canvas that allows you to move on
    game2played = true;
}
```

Station 3

- This section will detail all code pertaining to STN 3

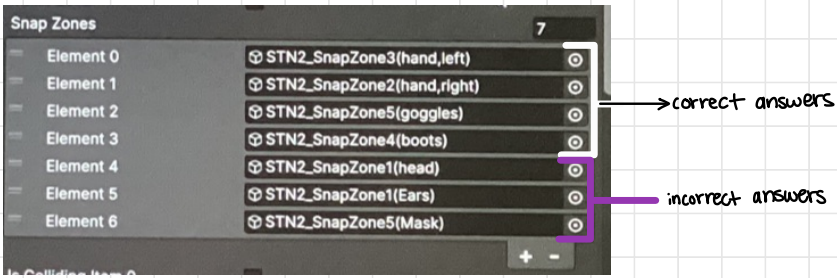
1) restriction function in universal functions script

```
public void Station3Restrict()
{
    //called when button clicked at end of station2
    Debug.Log("stn3 restrics");
    //called when button clicked at end of station2
    game2over = true;
    mainCamera.transform.position = new Vector3(-35, 0, -40);
    mainCamTransform.localRotation = Quaternion.identity; // Reset local rotation
    Quaternion newRotation = Quaternion.Euler(0f, -90f, 0f);
    mainCamTransform.rotation = newRotation;
    stn3Effect.Play();
    // Start coroutine to stop the particle system after 10 seconds
    if (stopParticleCoroutine != null)
    {
        StopCoroutine(stopParticleCoroutine);
    }
    stopParticleCoroutine = StartCoroutine(StopParticleSystemAfterDelay(10f, stn3Effect));
}
```

// sets new camera rotation

2) Update function in universal functions Script (part 1, lines 185- 218)

```
185        // RESTRICTIONS FOR PPE SECTION----------------------------------------------------------------------
186        if (snapZones.Length > 0)
187        {
188            // FOR PPE SECTION, check if the correct items are colliding and input answer into a boolean
189
190            //LEFT GLOVE                                                snapzones are in an array to hold the elements
191            XRSocketInteractor firstSnapZoneInteractor = snapZones[0].GetComponent<XRSocketInteractor>();
192            if (firstSnapZoneInteractor != null)
193            {
194                isCollidingItem0 = CheckCollidingWithTag(firstSnapZoneInteractor, "GloveLeft");
195                // Check if first snapzone in array is colliding w/correct object (in this case it's the left glove)
196            }
197            //RIGHT GLOVE
198            XRSocketInteractor secondSnapZoneInteractor = snapZones[1].GetComponent<XRSocketInteractor>();
199            if (secondSnapZoneInteractor != null)
200            {                 Boolean to flag if correct object is colliding with correct zone
201                isCollidingItem1 = CheckCollidingWithTag(secondSnapZoneInteractor, "GloveRight");
202                // if (isCollidingItem1) { DisplayMessage("glove R collide correct"); }
203            }
204            //GOGGLES
205            XRSocketInteractor thirdSnapZoneInteractor = snapZones[2].GetComponent<XRSocketInteractor>();
206            if (thirdSnapZoneInteractor != null)
207            {
208                isCollidingItem2 = CheckCollidingWithTag(thirdSnapZoneInteractor, "Goggles");
209                // if (isCollidingItem2) { DisplayMessage("goggles collide correct"); }
210            }
211            //BOOTS
212            XRSocketInteractor fourthSnapZoneInteractor = snapZones[3].GetComponent<XRSocketInteractor>();
213            if (fourthSnapZoneInteractor != null)
214            {
215                isCollidingItem3 = CheckCollidingWithTag(fourthSnapZoneInteractor, "Boots");
216                // if (isCollidingItem3) { DisplayMessage("boots collide correct"); }
217            }
218        }
```



- There are 4 correct PPE items that need to be in their correct places
- each correct item has a boolean variable that flags if it is in the correct spot.

example:

```
//GOGGLES
XRSocketInteractor thirdSnapZoneInteractor = snapZones[2].GetComponent<XRSocketInteractor>();
```

- Set the variable thirdSnapZoneInteractor which is of type xRSocketInteractor and set it to the xR socket interactor component of the 3rd snapzone in the array (element 2).

```
if (thirdSnapZoneInteractor != null)
{
    isCollidingItem2 = CheckCollidingWithTag(thirdSnapZoneInteractor, "Goggles");
    // if (isCollidingItem2) { DisplayMessage("goggles collide correct"); }
}
```

- a boolean variable "isCollidingItem2" then is set depending on if the correct snapzone is colliding with the correctly tagged object.

3) Check Colliding with Tag function in universal functions Script

```
626     bool CheckCollidingWithTag(XRSocketInteractor interactor, string tag)
627     {
628         // Check if there is any interactable currently selected in the interactor
629         if (interactor.selectTarget != null)
630         {
631             XRBaseInteractable selectedInteractable = interactor.selectTarget;
632
633             // Check if the interactable has a collider and is tagged with the specified tag
634             Collider interactableCollider = selectedInteractable.GetComponent<Collider>();
635             if (interactableCollider != null && interactableCollider.CompareTag(tag))
636             {
637                 // DisplayMessage("full is full");
638                 return true;
639             }
640         }
641         else
642         {
643             return false;
644         }
645         return false;
646     }
```

a)

```
626     bool CheckCollidingWithTag(XRSocketInteractor interactor, string tag)
```

→ means the function returns a Boolean value

b)

```
XRBaseInteractable selectedInteractable = interactor.selectTarget;
```

- The selected interactible object from the 'interactor' is assigned to a new variable 'selected Interactible' of type XRBaseInteractible.

c)

```
// Check if the interactable has a collider and is tagged with the specified tag
Collider interactableCollider = selectedInteractable.GetComponent<Collider>();
```

- retrieves the collider component attached to the 'selectedInteractable' object.

- Function exits :

TRUE - if the correctly tagged object is in it.

FALSE - if the wrong object is in it.

3) Update function in universal functions Script (part 2 lines 219-224)

```
219     //check if any of the zones that are supposed to be empty are colliding
220     bool EmptySnapZonesColliding = AreEmptySnapZonesColliding(4, 6);
221     // assigns a bool to flag if any of the zones that should be empty, are colliding with any object.
222     //CHECK RESTRICTIONS
223     * we pass through values 4 & 6 to signal the start and end of the array values that should be empty
224
```

4) Are Empty Snap Zones Colliding function in Universal Functions Script

```
1 reference
bool AreEmptySnapZonesColliding(int startIndex, int endIndex)
{
    for (int i = startIndex; i <= endIndex; i++)
    {
        XRSocketInteractor snapZoneInteractor = snapZones[i].GetComponent<XRSocketInteractor>();
        if (snapZoneInteractor.selectTarget != null)
        {
            // DisplayMessage("empty is colliding");
            return true; // Return true as soon as we find a colliding snap zone
        }
    }
    // DisplayMessage("no wrong collisions");
    return false; // Return false if none of the snap zones are colliding
}
```

(4) (6)

gets the interactor component for each socket interactor as it runs the loop.

5) Update function in universal functions Script (part 3 lines 225-268)

Part (a) lines 232-266

```
232     if (!unrestrict2 && game2over) //station2 is done (unrestrict1 = true) but station3 has not been completed (unrestrict2 = false)
233     {
234         Debug.Log("stn3 restricts");
235         if (EmptySnapZonesColliding || !isCollidingItem0 || !isCollidingItem1 || !isCollidingItem2 || !isCollidingItem3)
236         {
237             DisplayMessage("stn3restrictON");
238             HideCanvas(stn3END);
239             if (currentPosition.x < minXLimit2)
240             {
241                 currentPosition.x = minXLimit2;
242                 mainCamera.transform.position = currentPosition;
243             }
244             else if (currentPosition.x > maxXLimit2)
245             {
246                 currentPosition.x = maxXLimit2;
247                 mainCamera.transform.position = currentPosition;
248             }
249
250             if (currentPosition.z > maxZLimit2)
251             {
252                 Debug.Log("max z lim2 called");
253                 currentPosition.z = maxZLimit2;
254                 mainCamera.transform.position = currentPosition;
255             }
256             else if (currentPosition.z < minZLimit2)
257             {
258                 currentPosition.z = minZLimit2;
259                 mainCamera.transform.position = currentPosition;
260             }
261         }
262         else
263         {
264             EnableCanvas(stn3Correct);
265             GameOver3();
266         }
```

false while the snapzones are not colliding w/ correct object

True if a snapzone is not empty when it should be.

Restrictions Enacted

The code only enters the "else" loop if the correct objects are in their correct spots, and no incorrect PPE is being used.

part (b) lines 225 - 231

```
225         if ((EmptySnapZonesColliding || !isCollidingItem0 || !isCollidingItem1 || !isCollidingItem2 || !isCollidingItem3) && unrestrict2 && game2over)
226         {
227             DisplayMessage("restrictions should be on but game over has been called already");
228             HideCanvas(stn3Correct);
229             HideCanvas(stn3END);
230             unrestrict2 = false; //resets game over
231         }
```

- this 'if' loop is almost the same as the one in part (a) however it includes Unrestrict 2 and game2over these booleans are changed when game 3 over is called at the end of part (a) to signal that the user has created a correct combination.
- this loop is needed incase the user changes their answer specifically from the correct answer to an incorrect one. Once a correct answer is accounted for GameOver3(); function is called. So any changes to your answer after it is called would not be accounted for. Therefore this extra clause is needed to re-introduce the restrictions if your answer changes.

b) game over 3 function in the universal functions script.

```
1 reference
public void GameOver3()
{
    [redacted]
    unrestrict2 = true; // stops restrictions
    EnableCanvas(stn3END);

    mainCanvasText.text = "Congratulations you have completed station 3, please progress to station 4!\n" +
                          "here are your scores so far\n" +
                          $"Station 1: You correctly guessed 9/9 labels, you had {pointCounterStation1Script.wrong}  wrong attempts" + "\n Station 2: you correctly matched
    game3played = true;
}
1 reference
```

Station 4

- This section will detail all code pertaining to STN 4

1) Station 4Restrict in Universal Functions Script

```
public void Station4Restrict()
{
    //called when button clicked at end of station3
    Debug.Log("stn4 restrics");
    game3over = true;
    mainCamera.transform.position = new Vector3(5, 0, -43);
    mainCamTransform.localRotation = Quaternion.identity; // Reset local rotation
    Quaternion newRotation = Quaternion.Euler(0f, -180f, 0f); // New rotation of Camera
    mainCamTransform.rotation = newRotation;
    stn4Effect.Play();
    // Start coroutine to stop the particle system after 10 seconds
    if (stopParticleCoroutine != null)
    {
        StopCoroutine(stopParticleCoroutine);
    }
    stopParticleCoroutine = StartCoroutine(StopParticleSystemAfterDelay(10f, stn4Effect));
}
```

2) Update function in the Universal Functions Script

```
//RESTRICTIONS FOR DROPDOWN-----------------------------
if (game3over && !unrestrict3)
{
    DisplayMessage("stn 4 restrct");
    if (currentPosition.x < minXLimit3)
    {
        DisplayMessage("out of rnge");
        currentPosition.x = minXLimit3;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.x > maxXLimit3)
    {
        DisplayMessage("out of rnge");
        currentPosition.x = maxXLimit3;
        mainCamera.transform.position = currentPosition;
    }

    if (currentPosition.z > maxZLimit3)
    {
        DisplayMessage("out of rnge");
        currentPosition.z = maxZLimit3;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.z < minZLimit3)
    {
        Debug.Log("hi");
        currentPosition.z = minZLimit3;
        mainCamera.transform.position = currentPosition;
    }
}
if (greenFlag.greenFlag && !game4isOverFcnCalled)
{
    game4isOverFcnCalled = true;
    GameOver4();
}
```

Restrict movement

Syntax: variable for scriptname. value you want to access

enters loop when correct answer has been selected, flag to ensure this function is called only once

3) FirstAid.CS (Script for Station 4)

```
public class FirstAid : MonoBehaviour
{
    public TMP_Dropdown[] dropdowns; // Array of dropdown UI elements
    public GameObject panelToColor; // Reference to the UI panel to change color
    public TMP_Text feedbackText;
    public GameObject mainCamera;// reference to main camera
    public UniversalFunctions UniversalFunctionsScript;
    private Image panelImage; // Reference to the Image component of the panel
    public bool greenFlag = false;

    Unity Message | 0 references
    void Start()
    {
        // Get the Image component from the panelToColor GameObject
        panelImage = panelToColor.GetComponent<Image>();

        // call the function for each options OnChangedEvent
        for (int i = 0; i < dropdowns.Length; i++)
        {
            int index = i; // Store the current index in a local variable to avoid closure issues
            dropdowns[i].onValueChanged.AddListener(delegate {
                CorrectAnswerSelected(index, dropdowns[index].value);
            });
        }
    }
```

Loop is used to set up listners for each each dropdown option

used to determine when an option has been selected and what to do when it is selected.

```
    public void CorrectAnswerSelected(int dropdownIndex, int optionIndex)
    {
        // Check if the dropdown option index is 1 (Correct answer)
        if (optionIndex == 1)
        {
            greenFlag = true;
            // Change the color of the panel to green
            panelImage.color = Color.green;
            // Update feedback text
            feedbackText.text = "You selected the correct answer!";
        }
        else
        {
            // display answer as wrong, keep original message
            panelImage.color = Color.red;
        }
    }
}
```

the index of the dropdown that triggered the method to be called.

4) gameover4 function in universal functions script

```
public void GameOver4()
{
    unrestrict3 = true;
    EnableCanvas (stn4END);
    mainCanvasText.text = "Congratulations you have completed station 4, please progress to station 5!\n" +
                          "here are your scores so far\n" +
                          $"Station 1: You correctly guessed 9/9 labels, you had {pointCounterStation1Script.wrong}  wrong attempts" +
                          "\n Station 2: you correctly matched 3/3 of the SDS labels" +
                          "\n Station 3: you correctly matched all the PPE! The correct answer was: Goggles, Gloves, and Boots." +
                          "\n Station 4: you selected the correct answer to the first aid question!";
    game4played = true;

}
```

Station 5

- This section will detail all code pertaining to STN 5

1) Station 5 Restrictions function in the Universal functions Script

```
    public void Station5Restrict()
    {
        // called when button clicked at end of stnr
        //Debug.Log("stn4 restrics");
        game4over = true;
        mainCamera.transform.position = new Vector3(40, 0, -5); //Transport to new start position
        mainCamTransform.localRotation = Quaternion.identity; // Reset local rotation
        Quaternion newRotation = Quaternion.Euler(0f, 90f, 0f); // New Rotation Position
        mainCamTransform.rotation = newRotation;
        stn5Effect.Play();
        // Start coroutine to stop the particle system after 10 seconds
        if (stopParticleCoroutine != null)
        {
            StopCoroutine(stopParticleCoroutine);
        }
        stopParticleCoroutine = StartCoroutine(StopParticleSystemAfterDelay(10f, stn5Effect));
```

2) Update function in universal functions Script

```
//RESTRICTIONS FOR FILL IN THE BLANKS--------------------
if (game4over && !unrestrict5)
{
    Debug.Log("stn 5 restrics");


    if (currentPosition.x < minXLimit4)
    {
        currentPosition.x = minXLimit4;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.x > maxXLimit4)
    {
        [redacted]
        currentPosition.x = maxXLimit4;
        mainCamera.transform.position = currentPosition;
    }

    if (currentPosition.z > maxZLimit4)
    {
        currentPosition.z = maxZLimit4;
        mainCamera.transform.position = currentPosition;
    }
    else if (currentPosition.z < minZLimit4)
    {
        [redacted]
        currentPosition.z = minZLimit4;
        mainCamera.transform.position = currentPosition;
    }
}
```

Restrictions for movement

3) Static Script, attached to each TMP Input element on the Canvas

```
public class Static : MonoBehaviour
{
    public string correctWord; // Correct word for this input field

    private TMP_InputField inputField;
    private bool isCorrectWordEntered = false; // Flag to track if the correct word has been entered

    Unity Message | 0 references
    private void Start()
    {
        inputField = GetComponent<TMP_InputField>(); //retrieves input field component from gameobject.

        if (inputField == null)
        {
            Debug.LogError("TMP_InputField component not found on this GameObject.");
            return;
        }

        // Subscribe to TMP input field events
        inputField.onSelect.AddListener(OnInputFieldSelected);
        inputField.onSubmit.AddListener(OnInputFieldSubmitted);
```

subscribes to onSelect and onSubmit to handle user interactions

↳ not as relevant, method is only used to debug.

```
        // Subscribe to the OnTextSubmitted event of NonNativeKeyboard
        if (NonNativeKeyboard.Instance != null)
        {
            NonNativeKeyboard.Instance.OnTextSubmitted += OnTextSubmittedHandler; // subscribes to keyboards ontext submitted
            NonNativeKeyboard.Instance.SubmitOnEnter = true; // Set SubmitOnEnter to true to trigger text submission on "Enter"
        }
        else
        {
            Debug.LogError("NonNativeKeyboard.Instance is null. Make sure it's correctly set up in your scene.");
        }
    }
```

```
private void OnInputFieldSelected(string selectedText)
{
    if (isCorrectWordEntered)
    {
        // Prevent text editing if the correct word has been entered
        inputField.text = correctWord;
        return;
    }

    // Ensure NonNativeKeyboard instance is available
    if (NonNativeKeyboard.Instance != null)
    {
        // Clear the text in the input field
        inputField.text = string.Empty;

        // Assign the current TMP_InputField when selected
        NonNativeKeyboard.Instance.InputField = inputField;

        // Present the keyboard with the cleared value from input field
        NonNativeKeyboard.Instance.PresentKeyboard(inputField.text);
    }
    else
    {
        Debug.LogError("NonNativeKeyboard.Instance is null. Make sure it's correctly set up in your scene.");
    }
}
```

adding "return;" at the end of the 'if' block means that if this block is entered, the method exits at the end of it and will not execute the rest of the code in the function.

'OnInputFieldSelected' Method:

- Called when input field is selected
- if correct word has been entered ('isCorrectWordEntered' = True), it keeps the field as the correct word and does not allow further editing
- if correct word has <u>not</u> been entered ('isCorrectWordEntered' = False), it clears the input field and presents the keyboard.

```
private void OnTextSubmittedHandler(object sender, EventArgs e)
{
    if (sender is NonNativeKeyboard keyboard) // handles the text submission from "NonNative keyboard"
    {
        string submittedText = keyboard.InputField.text.ToLower(); // sets all text to Lowercase

        // Check if the submitted text matches the correct word
        if (submittedText == correctWord.ToLower())
        {
            ChangeInputFieldColor(Color.green);
            isCorrectWordEntered = true; // Set the flag to indicate the correct word has been entered

            // Prevent further changes to the input field text
            inputField.text = correctWord;
            inputField.interactable = false; // Disable further text input
        }
        else
        {
            ChangeInputFieldColor(Color.white);
        }
    }
}
```

'onTextSubmittedHandler' Method:

- compares submitted text with 'correctWord'.
  - ↳ if they match, sets field to green and isCorrectWordEntered to 'True'.
  - ↳ if they do not match it resets input field to white

```
2 references
private void ChangeInputFieldColor(Color color)
{
    // Method to change the color of the input field
    if (inputField != null && inputField.image != null)
    {
        inputField.image.color = color;
    }
}

Unity Message | 0 references
private void OnDestroy()
{
    // Clean up event listeners to prevent memory leaks
    if (inputField != null)
    {
        inputField.onSelect.RemoveListener(OnInputFieldSelected);
        inputField.onSubmit.RemoveListener(OnInputFieldSubmitted);
    }

    if (NonNativeKeyboard.Instance != null)
    {
        NonNativeKeyboard.Instance.OnTextSubmitted -= OnTextSubmittedHandler;
        NonNativeKeyboard.Instance.InputField = null; // Clear InputField reference
    }
    }
}
}
```

unsubscribes from events to prevent memory leaks

↳ clears references.

4) Check if green Script (each TMP element has their own attached)

```
using System.Collections.Generic;
using UnityEngine;
using TMPro;

Unity Script (2 asset references) | 1 reference
public class CheckIfGreen : MonoBehaviour
{
    public TMP_InputField inputField; // Reference to TMP InputField assigned in the Inspector
    public bool isGreenIgnition = false;
    public UniversalFunctions UniversalFunctionsScript;

    Unity Message | 0 references
    void Update()
    {
        // Check if the input field's color has changed to green
        CheckInputFieldColor();
    }

    1 reference
    private void CheckInputFieldColor()
    {
        if (inputField != null && inputField.image != null)
        {
            if (inputField.image.color == Color.green && !isGreenIgnition)
            {
                isGreenIgnition = true; // Set the flag to true once the color turns green
                if (UniversalFunctionsScript != null)
                {
                    UniversalFunctionsScript.HandleGreenColor(); // Call a method or perform an action when the color turns green
                }
            }
        }
    }
}
```

// checks if the field is green and makes sure the loop only enters once.

5) Handle Green Colour function in universal functions script

```
public void HandleGreenColor()
{

    if (checkIfGreenScript.isGreenIgnition &&
        checkIfGreenSpillageScript.isGreenSpillage &&
        checkIfGreenStaticScript.isGreenStatic &&
        checkIfGreenSlipperyScript.isGreenSlippery)
    {
        unrestrict5 = true;
        game5played = true;
        if (fillInBlanks)
        {
            mainCanvasText.text += "\n Station 5: you correctly filled in the blanks of the SDS labels, the answers were... 'ignition','static','slippery', and 'spillage'";
            fillInBlanks = false;
        }
        // Change color of Menu canvas when all conditions are met
        ChangeCanvasColor(fillInBlank, Color.green);
    }
}
```

enters if block only when all inputs have been turned green (ie. all are correct)

Final note:

- in update function of universal functions script...

```
        //END OF ENTIRE GAME-----------------------------------------------------------------------
        if (game1played && game2played && game3played && game4played && game5played)
        {
            // do stuff to indicate game is over....
            DisplayMessage("game over ;)");
            mainCamera.transform.position = new Vector3 (0, 0, 0);
            mainCamTransform.localRotation = Quaternion.identity; // Reset local rotation
            EnableCanvas(endGame);

        }
    }
    //END OF UPDATE FUNCTION-------------------------------------------------------------------
```