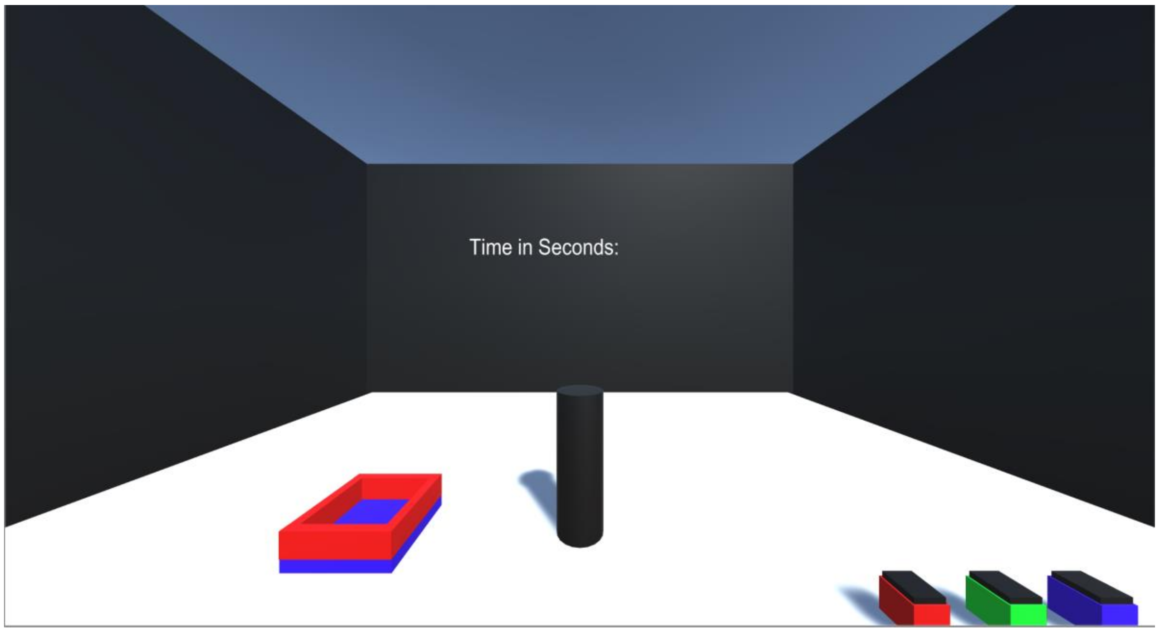
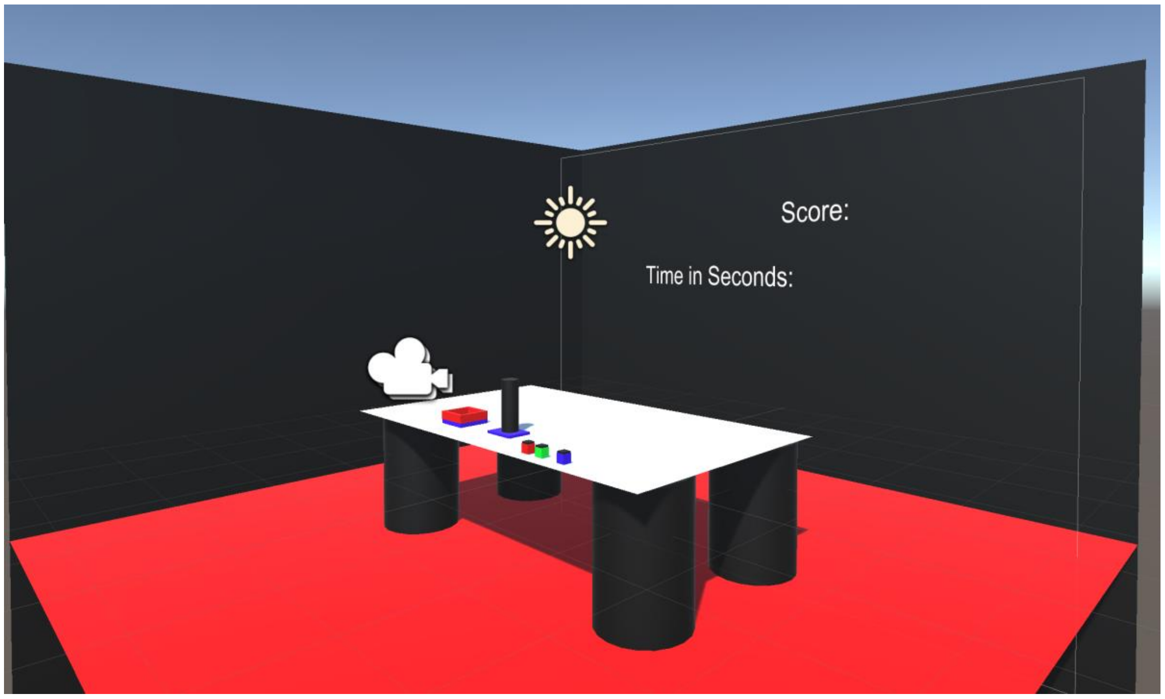
Assumption:
Already know how to integrate the Oculus SDK and Leap Motion SDK into Unity.

Project Information:

- Description of Project
    - My project was a virtual reality application that allowed for cognitive agency rehabilitation in patients through experiencing a gamified environment.
- How the project worked
    - A user will launch the application and will be put into an assessment level. The whole game revolves around a simple task: being able to pick up a bottle and place it in a coaster. The user will be placed within a box where the box will change colors from red to yellow to green. Once the box is green a timer will start measuring how long it takes for the user to place the bottle into the coaster. Within the assessment level, the user will try to place the bottle in the coaster as fast as he or she can. Based on the time it took the user to place the bottle into the coaster he/she will be assigned to a certain level that will require them to move faster to progress throughout the game.
- How does moving up levels work?
    - If the user gets placed into level 1 based on their performance in the assessment level, he/she will need to place the bottle in the coaster within 10 seconds. If the user manages to do that they will move to level 2 and the time required to place the cup in the coaster will decrease.
- How can you lose?
    - Yes, every time a user fails to complete a level, meaning they don't place the bottle in the coaster within a certain time they will lose a life and have to redo that level. If the user loses 3 lives he/she will lose the game.
- How Does Scoring Work
    - For every level, the user completes he/she will receive 100 points. If the user drops the bottle he/she will lose 50 points. If the user gets placed into level 7 automatically he/she will receive 700 points.
- How does this relate to cognitive agency?
    - The more times the user attempts to play this game, the faster the room surrounding the user will turn green(indicating for them to start to play). Overall, the more the user plays the more their sense of agency is increased.
- Picture of the Unity Environment

- ○ 

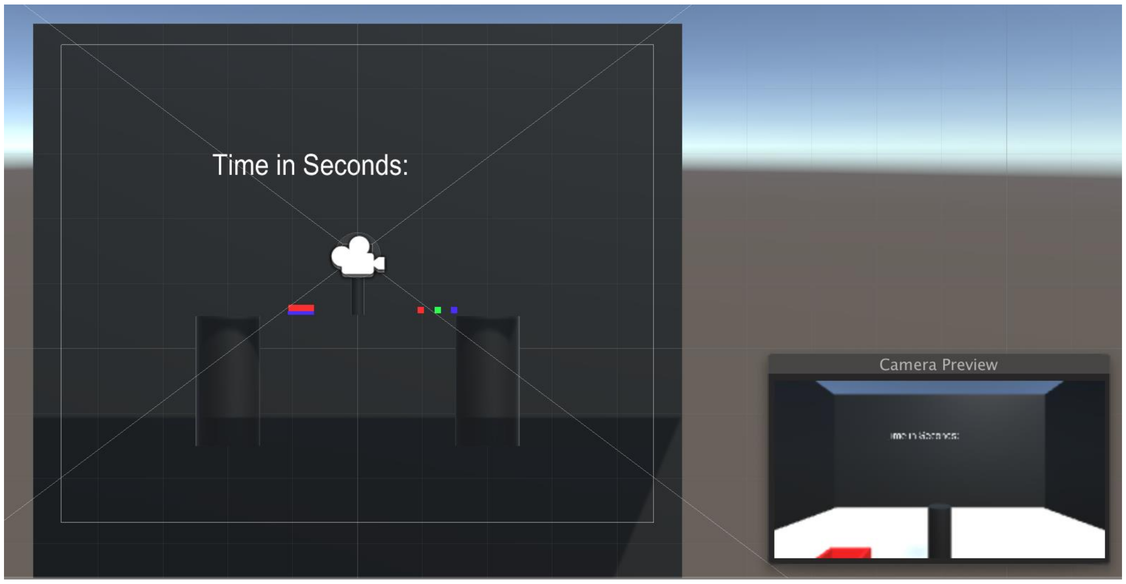
- ○ 

Creating a Level

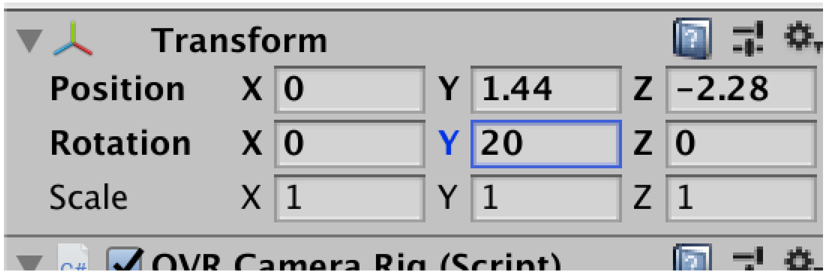
- Level UI
    - What Appears in a level:
        - The user is loaded into the scene and has 3 buttons to the right. One is to start the timer, one is to reset the position of the bottle if it gets knocked off the table and the last one is to stop the timer. After clicking the start timer button the button disappears so the user won't accidentally click it again. Then the user finishes placing the bottle in the cupholder and

clicks the stop timer button. This checks whether the bottle is in the coaster and whether to send the user to the next level. **Notice how the user has to move his head side to side to view all aspects of the scene. Therefore when the user starts the scene at 0 degrees and may end up x amount of degrees to the right. The camera position in upcoming levels should account for this change. Please look at the Setting Camera to fixed position guide to see how I tackled this problem.**

- Setting Camera to a fixed position
    - Creating an environment in unity revolves around what the user sees. This is where the camera element comes into play. The camera in unity allows you to depict the specific view of the scene in the eyes of the player. Especially when building a Virtual Reality scene your INITIAL camera position matters a lot.



    -
    - Between levels, as the user progresses the camera position resets the entire environment. For instance, a user can be in level 1 turned 45 degrees to the right and when they move onto level 2 the initial 45 degrees to the right is viewed as 0 degrees. You can't pass a camera through levels to account for the user constantly shifting between levels. If you face a similar problem to me, I ended up tilting the camera in levels 1-7 20 units for their value.

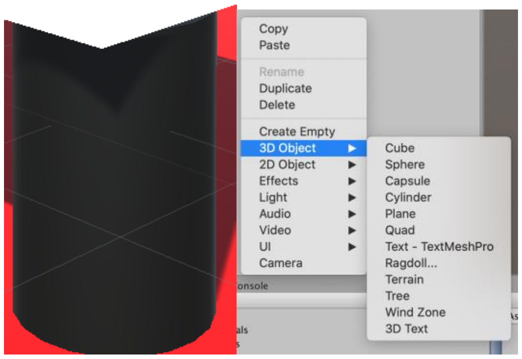


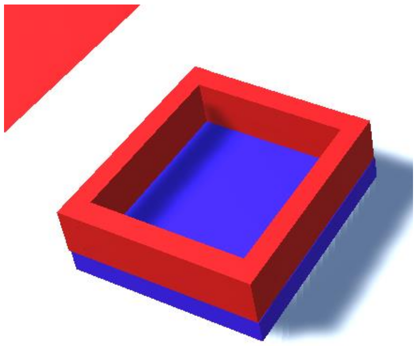
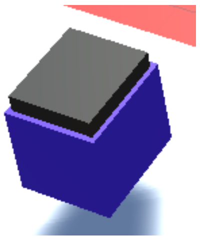
Transform

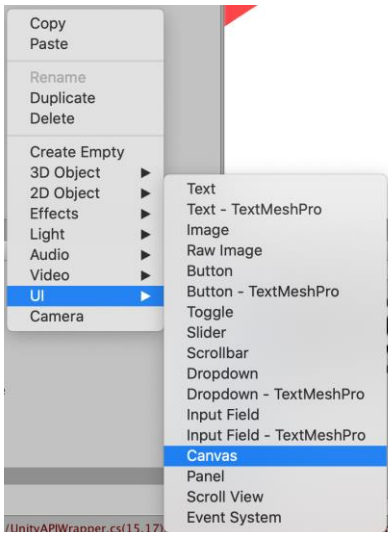
Position	X	0	Y	1.44	Z	-2.28
Rotation	X	0	Y	20	Z	0
Scale	X	1	Y	1	Z	1

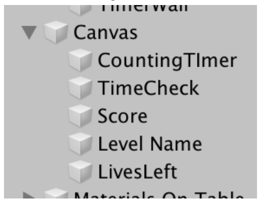
OVR Camera Rig (Script)

    -

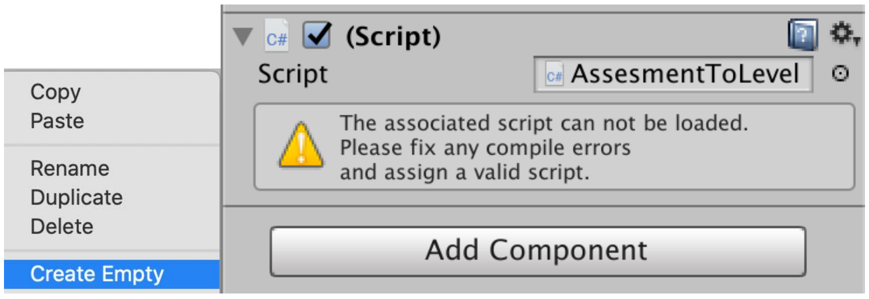
        - This accounts for the user being physically already turned to the right when they complete the previous levels. If this were not in place, the user will constantly be turning more and more to the right throughout the levels.
    - Fill in how you created User Interface
        - Legs
            - Go to the Hierarchy on the left-hand side and create a 3-D object called a sphere. Adjust the dimensions accordingly.
    - 

        - Table
            - Follow the above steps and create a Rectangle and adjust your x,y,z values to place the rectangle above your legs.
        - Coaster
            - 
            - Create a rectangle for the sides and place it on top of a rectangle on the bottom.
        - Button
            - 
            - These buttons have a special pressable action that I will go into more in-depth later on in this tutorial on how to use it. Feel free to take this asset from my code.

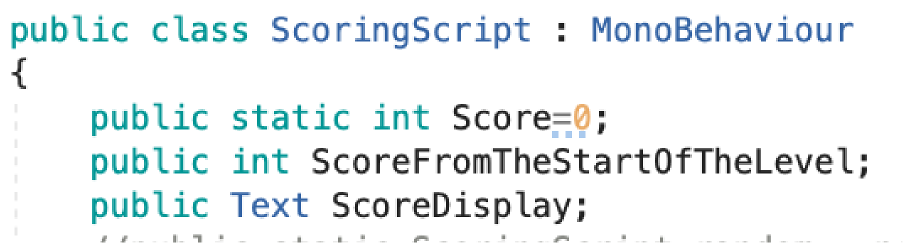
- Displaying Information to the user
    - Canvas
        - How to Create a Canvas
            - 

        - Scoring
            - Create a text element underneath the Canvas within the hierarchy and edit it.
            - 

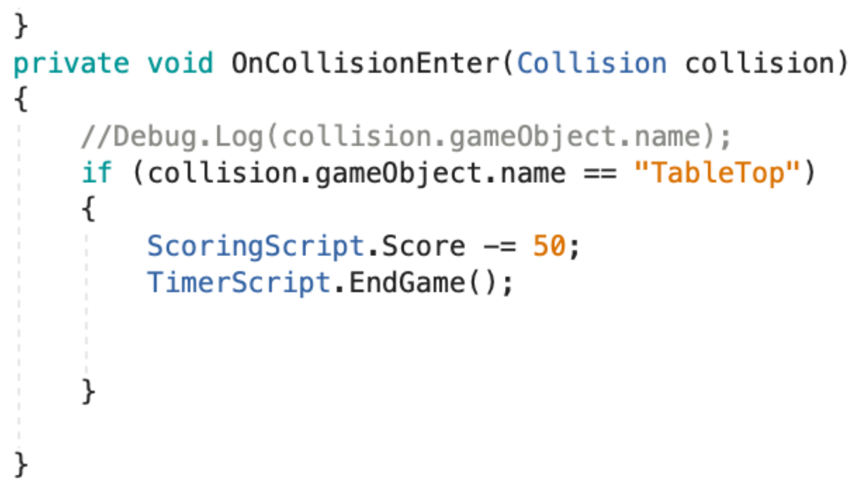
        - Time
            - Create a text element underneath the Canvas within the hierarchy and edit it.
- Backend Code
    - Enabling A Script
        - How does using scripts in C# work?
            - By this point in the tutorial, I am hoping you know how to read and write C# code. I am also assuming that you have the button object from my code example to reference.
            - Create an Empty Object in the scene you want to run your script then for that object add a script component. This will run your code as soon as the scene is loaded, so keep that in mind.



The associated script can not be loaded.
Please fix any compile errors
and assign a valid script.

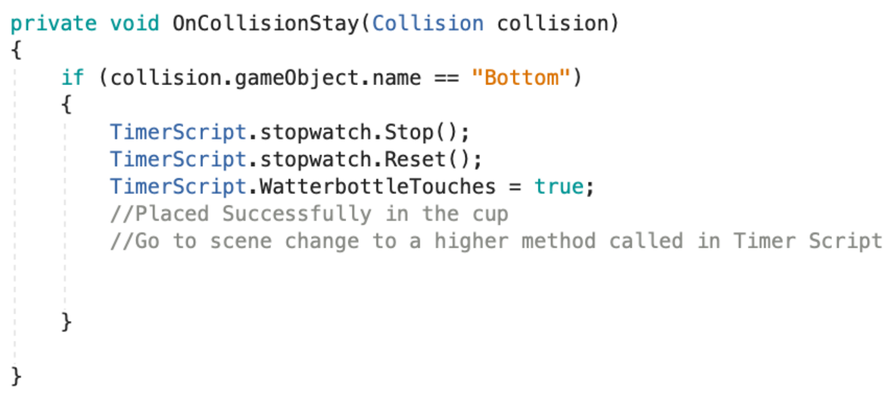
- Score Tracking
  - Increasing score when user completes levels
    - When you create a public variable in a unity script you will be able to give inputs from the game object.
    - 

```
public class ScoringScript : MonoBehaviour
{
    public static int Score=0;
    public int ScoreFromTheStartOfTheLevel;
    public Text ScoreDisplay;
```

    - For instance, the variable ScoreFromTheStartOfTheLevel was assigned per level in the game object that I called the script in. This allowed me to reuse the same script for each level but only change the score value in the unity object. This allows me to know what to add to the total score when the user completes a level.
  - Decreasing the score when the user drops the bottle
    - 

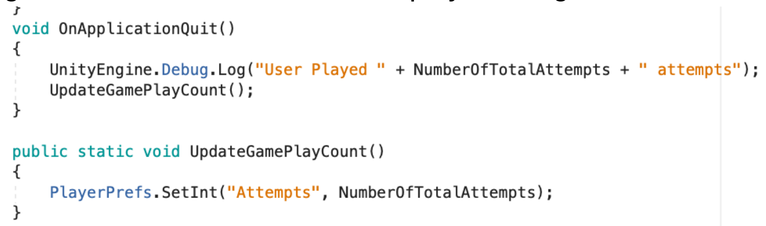
```
}
private void OnCollisionEnter(Collision collision)
{
    //Debug.Log(collision.gameObject.name);
    if (collision.gameObject.name == "TableTop")
    {
        ScoringScript.Score -= 50;
        TimerScript.EndGame();


    }

}
```

    - I use the OnCollisionEnter method on the water bottle to check if it ever hits the tabletop, if it does I deduct 50 points from the score.
    - https://www.youtube.com/watch?v=bh9ArKrPY8w reference this video.
- Figuring out when the user placed the bottle inside the coaster



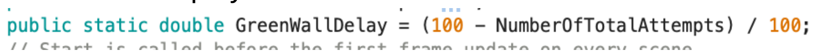
```
private void OnCollisionStay(Collision collision)
{
    if (collision.gameObject.name == "Bottom")
    {
        TimerScript.stopwatch.Stop();
        TimerScript.stopwatch.Reset();
        TimerScript.WatterbottleTouches = true;
        //Placed Successfully in the cup
        //Go to scene change to a higher method called in Timer Script


    }

}
```

        -
        - I had used the same collision feature that detects whether the bottle is touching the tabletop, however, this time I did it with the bottom of the coaster. In this scenario, the coaster object name is "Bottom."
- Integrating both Code and UI
    - How to integrate different levels and assign different information per level
        - As I mentioned above, if you make a value in your code public you will be able to assign it input from the unity object that you connected the script to.
        - https://www.youtube.com/watch?v=ck6OyNBC95c
            - Reference how the developer now can pass a specific score text value for the current scene he is developing. When used properly this should allow you to reuse the scripts you make and just change the values assigned for different levels.
    - How I enabled cognitive agency regeneration.
        - How do I track how many times the user has played the game?
            - As I mentioned earlier I tracked the amount of time the user played the game to speed up how fast the initial timer starts. The more the user played the game the faster the timer would start every round and the faster the user had to begin the task of picking up the bottle and placing it within the coaster.
                - Tracking the number of times the user played the game



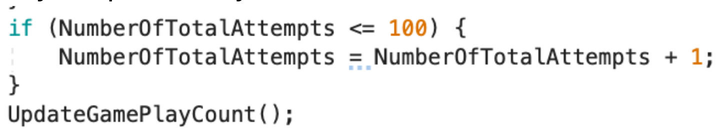
```
void OnApplicationQuit()
{
    UnityEngine.Debug.Log("User Played " + NumberOfTotalAttempts + " attempts");
    UpdateGamePlayCount();
}

public static void UpdateGamePlayCount()
{
    PlayerPrefs.SetInt("Attempts", NumberOfTotalAttempts);
}
```

                    -
                    - I created a file within the project that I would write to after the application was quit. So essentially every time the user played and finished the game I would count that as 1 run and store that on a local file within the project directory.

- Using this data to enhance the game
  - I would read the number of times the user played the game when the game initially was loaded and I would adjust the delay the green screen came/ when the game started accordingly to how many times the user played.
  - 

```
public static double GreenWallDelay = (100 - NumberOfTotalAttempts) / 100;
```

  - The greater times the user played (till 100 times played) the smaller and smaller the delay was. Once the user reached 100 times, I would just permanently keep the delay at 0.



```
if (NumberOfTotalAttempts <= 100) {
    NumberOfTotalAttempts = NumberOfTotalAttempts + 1;
}
UpdateGamePlayCount();
```

-