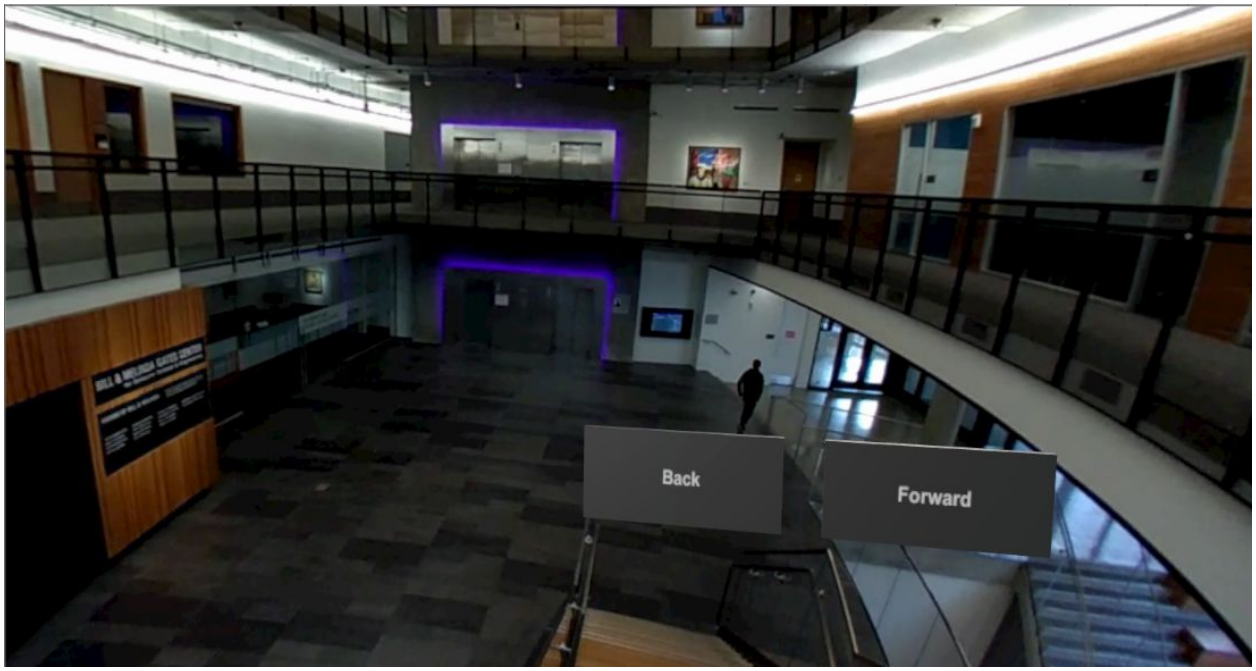
We split up work this week to finish our scene breakdown, reaching out to homeless shelters, Unity scene skeleton, and interactive rendered scene of sorting through items in a bin at the homeless shelter. Erika and Amy sent emails to a total of eight homeless shelters. So far we haven't gotten any positive responses, but we will reach out to more people next week. Amy and Alison worked on the rendered bin scene. Erika and Aaron worked on the Unity scene skeleton. Each of us worked on one scene in the **Scene Breakdown** (https://docs.google.com/presentation/d/1X4vrM9XZ-OF5aJUrj99OBIFz7SWxyYZOl-jiY7Bhjn4/edit?usp=sharing), in which we specified the camera view and dialogue for each of our different scenes.

We finished a very basic scene skeleton in Unity. We made a Unity scene for each different video we plan to film, named it accordingly, and added button transitions between each scene in the order they will actually need to transition. For now we're using placeholder videos in all of the scenes. Once we have the videos we will just need to add it to our Unity project and change one field!

Below is an example of one of the scenes:

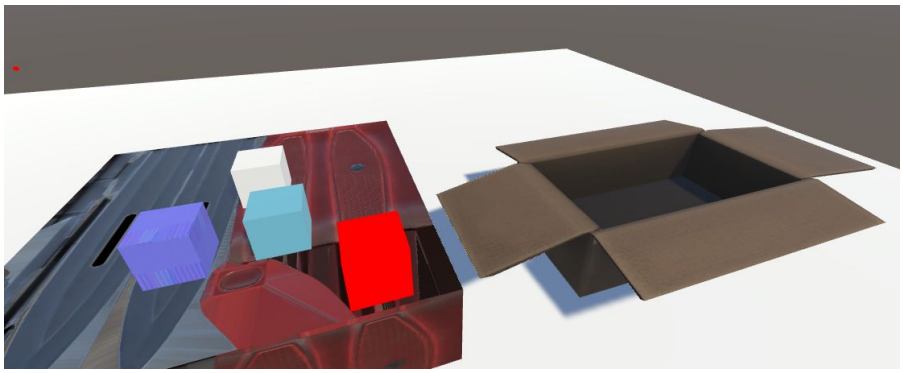


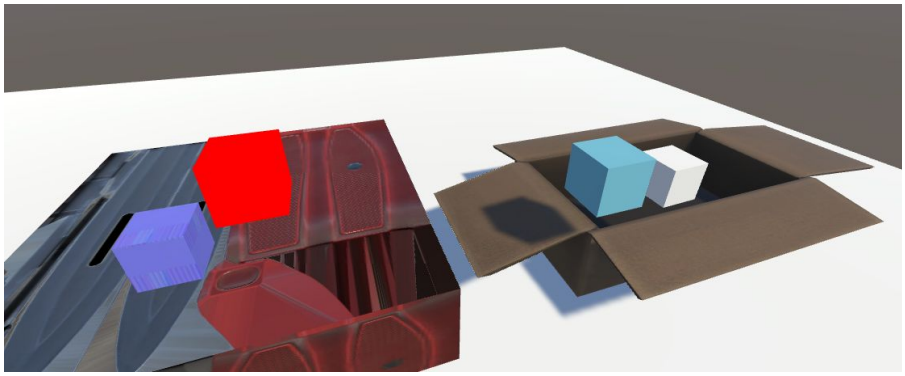
The code is not very interesting, but the relevant commit is **here** (https://github.com/UWRealityLab/vrcapstone19sp-team6/commit/8aeecc7e0741e6b69ebb35297c3811cd40dba8b1).

For our first experience of waking up in a shelter we are working to create the interactive portion of sorting through items in a bin that will be a rendered element with 3D background. We found it difficult to reach into the box and grab items so we decided to use ray casting with laser pointers to simulate the hand movements and be able to grab items easier. We simulated the

hand action such as hand hover on the pointer click such that a user can click on a object to “select” it and attach to hand and click again to detach. However, upon adding any physics components or interactable scripts such as rigid body the object isn’t actually able to attach to the hand/laser properly. We do need the objects to react correctly to gravity, and we suspect that the laser pointer script we wrote doesn’t work well with the hand/hover motions that are the default method to grab. We also spent time trying to debug this issue with John, however we weren’t able to find a solution to this problem yet, and are slightly blocked. We plan to continue looking into other methods of ray casting that work better with the hand and following up with John.

Here is an example of the placing the items into the box, and being able to interact with the items in the box to check if any items are missing. We have also included snippets of our code below where the first snippet is a wrapper on the SteamVR Laser Pointer, and the second snippet is a script used on our game objects to detect when to attach to the hand/laser.





```
0 references | 0 changes | 0 authors, 0 changes
public class LaserPointerWrapper : MonoBehaviour
{
    private SteamVR_LaserPointer pointer;
    public bool attached;

    [EnumFlags]
    public Hand.AttachmentFlags attachmentFlags = Hand.AttachmentFlags.ParentToHand | Hand.AttachmentFlags.DetachFromOtherHand;


    // Start is called before the first frame update
    0 references | 0 changes | 0 authors, 0 changes
    void Start()
    {
        pointer = gameObject.GetComponent<SteamVR_LaserPointer>();
        pointer.PointerIn += OnPointerIn;
        pointer.PointerOut += OnPointerOut;
        pointer.PointerClick += OnPointerClick;
        attached = false;
    }

    // Update is called once per frame
    0 references | 0 changes | 0 authors, 0 changes
    void Update()
    {

    }

    1 reference | 0 changes | 0 authors, 0 changes
    private void OnPointerIn(object sender, PointerEventArgs e)
    {
        IPointerEnterHandler pointerEnterHandler = e.target.GetComponent<IPointerEnterHandler>();
        //Debug.Log("entered onPointerIn");

        //pointerEnterHandler.OnPointerEnter(new PointerEventData(EventSystem.current));
        //e.target.gameObject.SendMessage("HandHoverUpdate", gameObject.GetComponent<Hand>());
    }

    1 reference | 0 changes | 0 authors, 0 changes
    private void OnPointerOut(object sender, PointerEventArgs e)
    {
        //Debug.Log("entered onPointerOut");
        IPointerExitHandler pointerExitHandler = e.target.GetComponent<IPointerExitHandler>();

        //pointerExitHandler.OnPointerExit(new PointerEventData(EventSystem.current));
        //e.target.gameObject.SendMessage("OnHandHoverEnd", gameObject.GetComponent<Hand>());
    }

    1 reference | 0 changes | 0 authors, 0 changes
    private void OnPointerClick(object sender, PointerEventArgs e)
    {
        Debug.Log("entered onPointerClick");
        if (attached)
        {
            Debug.LogWarning("DeAttaching " + e.target.gameObject);

            gameObject.GetComponent<Hand>().DetachObject(e.target.gameObject);
            attached = false;
        }
        else
        {
            Debug.LogWarning("Attaching " + e.target.gameObject);
            gameObject.GetComponent<Hand>().AttachObject(e.target.gameObject, GrabTypes.Trigger, attachmentFlags);
            attached = true;
        }
        //gameObject.GetComponent<Hand>().AttachObject(e.target.gameObject, GrabTypes.None, attachmentFlags);
        //e.target.gameObject.SendMessage("HandHoverUpdate", gameObject.GetComponent<Hand>());

    }
}
```

```
0 references | 0 changes | 0 authors, 0 changes
public class SimpleGrab : MonoBehaviour
{
    public Material grabMat;
    private Material originalMat;

    [EnumFlags]
    public Hand.AttachmentFlags attachmentFlags = Hand.AttachmentFlags.ParentToHand | Hand.AttachmentFlags.DetachFromOtherHand;

    0 references | 0 changes | 0 authors, 0 changes
    public void attachToHand(bool attached, GameObject go) {
        if (attached) {
            Debug.Log("attaching" + go);
            if (go.GetComponent<Hand>() != null) {
                go.GetComponent<Hand>().DetachObject(gameObject);
            }
        }
        else
        {
            if (go.GetComponent<Hand>() != null)
            {
                go.GetComponent<Hand>().AttachObject(gameObject, GrabTypes.None, attachmentFlags);
            }
        }
    }

     0 references | 0 changes | 0 authors, 0 changes
     private void HandHoverUpdate(Hand hand)
     {
        GrabTypes startingGrabType = hand.GetGrabStarting();
        Debug.Log("Entered HandHoverUpdate");
        hand.AttachObject(gameObject, GrabTypes.Trigger, attachmentFlags);
     }


    0 references | 0 changes | 0 authors, 0 changes
    private void HandAttachedUpdate(Hand hand)
    {
        if (hand.IsGrabEnding(gameObject))
        {
            hand.DetachObject(gameObject);
        }
    }
```

Next week we will film the park scene and the eviction scene. Erika and Aaron will work on the park scene and Amy and Alison will film the eviction scene. We hope to get them edited and put them into our Unity project as well! We will also send out emails to more people who may have contacts who would allow us to film briefly in a homeless shelter. Amy and Alison will also finish the bin interaction.

We are slightly blocked on this portion but we will continue to work with John to create the best user experience to interact with the items.We will also copy the staff on the emails we send this week in case that helps us get responses!