Escape the Room VR

Review: Objective

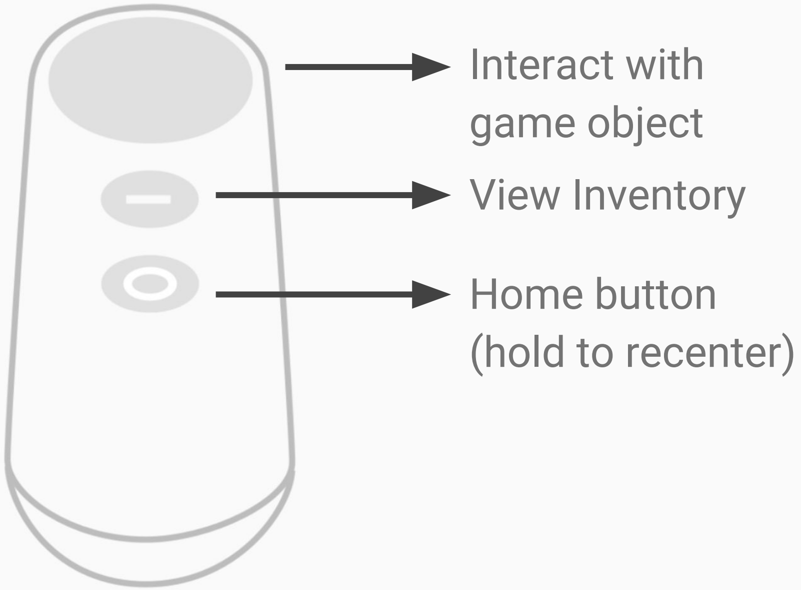
- **Goal:** build a virtual Escape the Room experience
- **Escape the Room**: a scenario where a group of people are locked in a room and must escape in a given amount of time

Review: Technologies

- Daydream virtual reality headset
- Android smartphone (e.g. Pixel / Pixel XL)
- Graphic shaders

Daydream VR & Android

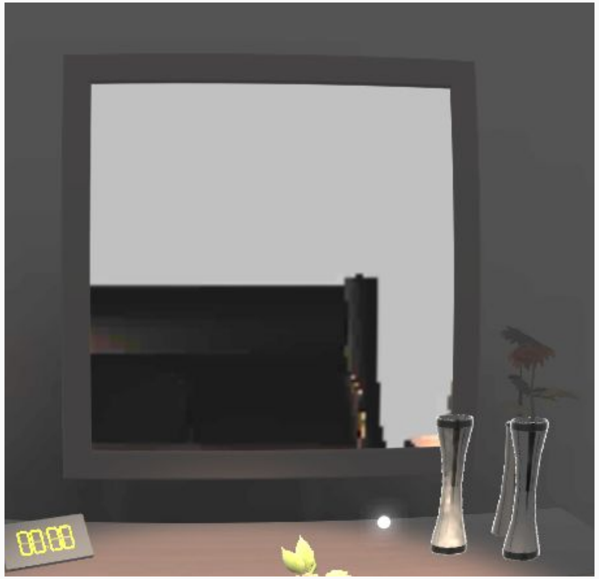
- The Daydream VR headset has:
  - Lenses
  - A slot to insert a Daydream supported Android smartphone
  - A controller that allows us to interact with the app



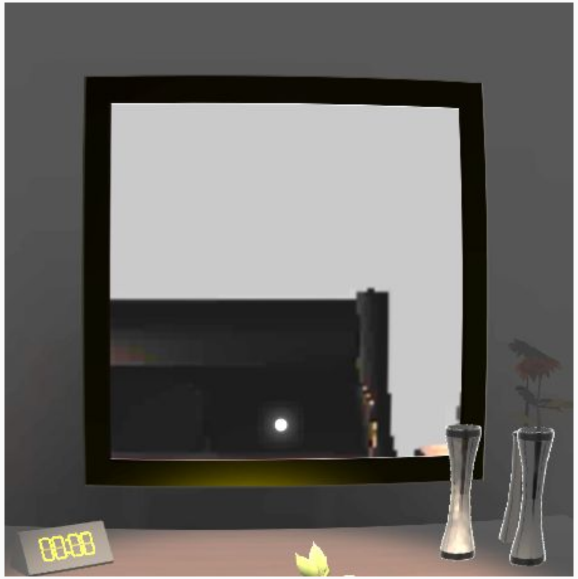
Graphic Shaders

- Highlighting game objects
  - Add a glow effect to distinguish interactable game objects in the scene
- Prevent object clipping
  - Allow specified UI objects to always be drawn over all other game objects
  - **Example:** the popup message describing the result of an interaction with a game object

Graphic Shaders



Observe the change of color with the glow effect on the mirror’s wooden frame.



Inspectable
Game Objects

```
public class Inspectable : MonoBehaviour {
    public string inspectMessage;

    public void Inspect() {
        // Show default observation text
        Inspect(inspectMessage);
    }

    public void Inspect(string msg) {
        // Show specific message in response to item use
        // Sends message to be deployed in a coroutine
        GameObject.Find("SubtitlesCanvas/MessageText")
            .GetComponent<ShowMessage>().RunMessage(msg);
    }
}
```

Collectable
Game Objects

```
public class Collectable : MonoBehaviour {
    public Item item; // The object to place in inventory
    private Inventory inventory;
    // Different materials based on controller’s pointer
    public Material inactiveMaterial, gazedAtMaterial;
    public UnityEvent dispatch; // Fire Callback

    void Awake() {
        inventory = FindObjectOfType<Inventory>();
        if (inventory == null) { … }
    }

    void Start() { … }

    public void Collect() {
        if (item) { /* Pick up item */ }
        if (dispatch != null) dispatch.Invoke();
    } // … more code follows
}
```

Inventory

View

```
public class Inventory : MonoBehaviour {
    private bool isVisible;
    public const int numItemSlots = 9;
    public Image[] itemImages = new
        Image[numItemSlots];
    public Item[] items = new Item[numItemSlots];
    public Item SelectedItem { get; private set; }

    void Start() {
        isVisible = false;
        SelectedItem = null;
    }

    void Update() {
        // Toggle inventory view on app button press
        if (GvrController.AppButtonUp) {
            ToggleVisibility();
        }
    } // … more code follows
}
```

Usable

Game Objects

```
public class Usable : Inspectable {
    public Item useItem;
    public string successMessage, failureMessage;
    public UnityEvent dispatch;  // Callback

    public void Use(Inventory inv) {
        // Check if item matches what's expected
        if (inv.SelectedItem == null) {
            Inspect(); // Item is not usable
        } else if (inv.SelectedItem == useItem) {
            Inspect(successMessage);
            inv.RemoveItem(useItem);
            if (dispatch != null) dispatch.Invoke();
        } else {
            Inspect(failureMessage);
        }
    }
}
```